WebSphere® eXtreme Scale Version 7.1
Administratorhandbuch

WebSphere eXtreme Scale Administratorhandbuch

Diese Ausgabe bezieht sich auf Version 7, Release 1 von WebSphere eXtreme Scale und, sofern in neuen Ausgaben nicht anders angegeben, auf alle nachfolgenden Releases dieses Produkts.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM WebSphere eXtreme Scale Version 7.1 Administration Guide,
herausgegeben von International Business Machines Corporation, USA

Herausgegeben von:
SW TSC Germany
Kst. 2877
August 2010

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellen

Informationen zum *Administratorhandbuch*

Der Dokumentationssatz zu WebSphere eXtreme Scale umfasst drei Handbücher, die die erforderlichen Informationen zur Verwendung des Produkts WebSphere eXtreme Scale, zur Programmierung für das Produkt und zur Verwaltung des Produkts enthalten.

Bibliothek von WebSphere eXtreme Scale

Die Bibliothek von WebSphere eXtreme Scale enthält die folgenden Bücher:

- Das *Administratorhandbuch* enthält die für Systemadministratoren erforderlichen Informationen, z. B. Planung von Anwendungsimplementierungen, Kapazitätsplanung, Installation und Konfiguration des Produkts, Starten und Stoppen von Servern, Überwachung der Umgebung und Sicherung der Umgebung.
- Das *Programmierhandbuch* enthält Informationen für Anwendungsentwickler zur Entwicklung von Anwendungen für WebSphere eXtreme Scale unter Verwendung der bereitgestellten API-Informationen.
- Das Handbuch *Produktübersicht* enthält einer Übersicht über die Konzepte von WebSphere eXtreme Scale, einschließlich Anwendungsfallszenarien und Lernprogrammen.

Zum Herunterladen der Handbücher rufen Sie die Bibliotheksseite von WebSphere eXtreme Scale auf.

Sie finden die Informationen aus dieser Bibliothek auch im Information Center von WebSphere eXtreme Scale.

Zielgruppe

Dieses Handbuch ist hauptsächlich für Systemadministratoren, Sicherheitsadministratoren und Systembediener bestimmt.

Aktualisierungen für dieses Handbuch

Sie erhalten Aktualisierungen zu diesem Handbuch, indem Sie die jeweils aktuelle Version des Handbuchs von der Bibliotheksseite von WebSphere eXtreme Scale herunterladen.

Hinweise zu Rückmeldungen

Wenden Sie sich an das Dokumentationsteam. Haben Sie die benötigten Informationen gefunden? Sind die Informationen präzise und vollständig? Senden Sie Ihre Kommentare zu dieser Dokumentation per E-Mail an wasdoc@us.ibm.com.

Kapitel 1. Einführung in WebSphere eXtreme Scale

Nach der Installation von WebSphere eXtreme Scale in einer eigenständigen Umgebung verwenden Sie die folgenden Schritte als einfache Einführung in die Funktionalität des Produkts als speicherinternes Daten-Grid.

Die eigenständige Installation von WebSphere eXtreme Scale enthält ein Muster, das Sie verwenden können, um Ihre Installation zu prüfen und sich mit der Verwendung eines einfachen eXtreme-Scale-Grids und -Clients vertraut zu machen. Das Einführungsmuster ist im Verzeichnis *Installationsstammverzeichnis*`/ObjectGrid/gettingstarted` enthalten.

Das Einführungsmuster ist eine Schnelleinführung in die Funktionen und die Basisoperationen von eXtreme Scale. Das Muster setzt sich aus Shell- und Stapelscripts zusammen, mit denen ein einfaches Grid ohne umfassende Anpassungen gestartet werden kann. Außerdem wird ein Clientprogramm, einschließlich des Quellcodes, bereitgestellt, mit dem Sie einfache Erstellungs-, Lese-, Aktualisierungs- und Löschoperationen (CRUD, Create, Read, Update, and Delete) für dieses Basis-Grid ausführen können.

Scripts und ihre Funktionen

Dieses Muster stellt die folgenden vier Scripts bereit:

Das Script `env.sh|bat` wird von den anderen Scripts aufgerufen, um die erforderlichen Umgebungsvariablen zu setzen. Normalerweise müssen Sie dieses Script nicht ändern.

- UNIX Linux `./env.sh`
- Windows `env.bat`

Das Script `runcat.sh|bat` startet den Katalogserviceprozess von eXtreme Scale auf dem lokalen System.

- UNIX Linux `./runcat.sh`
- Windows `runcat.bat`

Das Script `runcontainer.sh|bat` startet einen Containerserverprozess. Sie können dieses Script mehrfach mit jeweils eindeutigen Servernamen ausführen, um eine beliebige Anzahl an Containern zu starten. Diese Instanzen können zusammenarbeiten, um partitionierte und redundante Informationen im Grid zu speichern.

- UNIX Linux `./runcontainer.sh` *`eindeutiger_Servername`*
- Windows `runcontainer.bat` *`eindeutiger_Servername`*

Das Script `runclient.sh|bat` führt den einfachen CRUD-Client aus und startet die angegebene Operation.

- UNIX Linux `./runclient.sh` *`Befehl Wert1 Wert2`*
- Windows `runclient.sh` *`Befehl Wert1 Wert2`*

Für *Befehl* können Sie eine der folgenden Optionen einsetzen:

- Geben Sie `i` ein, um *Wert2* in das Grid mit dem Schlüssel *Wert1* einzufügen.

- Geben Sie `u` ein, um das Objekt mit dem Schlüssel *Wert1* in *Wert2* zu aktualisieren.
- Geben Sie `d` ein, um das Objekt mit dem Schlüssel *Wert1* zu löschen.
- Geben Sie `g` ein, um das Objekt mit dem Schlüssel *Wert1* abzurufen und anzuzeigen.

Anmerkung: Die Datei *Installationsstammverzeichnis*`/ObjectGrid/gettingstarted/src/Client.java` ist das Clientprogramm, das demonstriert, wie eine Verbindung zu einem Katalogserver hergestellt, eine ObjectGrid-Instanz abgerufen und die API "ObjectMap" verwendet wird.

Grundlegende Schritte

Verwenden Sie die folgenden Schritte, um das erste Grid zu starten und einen Client für die Interaktion mit dem Grid auszuführen.

1. Öffnen Sie eine Terminalsitzung oder ein Befehlszeilenfenster.
2. Verwenden Sie den folgenden Befehl, um zum Verzeichnis `gettingstarted` zu navigieren:

   `cd` *`Installationsstammverzeichnis`*`/ObjectGrid/gettingstarted`

   Ersetzen Sie *Installationsstammverzeichnis* durch den Pfad des Installationsstammverzeichnisses von eXtreme Scale bzw. durch den Stammdateipfad des *Installationsstammverzeichnisses* der extrahierten Testversion von eXtreme Scale.
3. Führen Sie das folgende Script aus, um einen Katalogserviceprozess auf dem lokalen Host (localhost) zu starten:
   - UNIX Linux `./runcat.sh`
   - Windows `runcat.bat`

   Der Katalogserviceprozess wird im aktuellen Terminalfenster ausgeführt.
4. Öffnen Sie eine weitere Terminalsitzung bzw. ein weiteres Befehlszeilenfenster, und führen Sie den folgenden Befehl aus, um eine Containerserverinstanz zu starten:
   - UNIX Linux `./runcontainer.sh server0`
   - Windows `runcontainer.bat server0`

   Der Containerserver wird im aktuellen Terminalfenster ausgeführt. Sie können die Schritte 5 und 6 wiederholen, wenn Sie weitere Containerserverinstanzen für die Unterstützung der Replikation starten möchten.
5. Öffnen Sie eine weitere Terminalsitzung bzw. ein weiteres Befehlszeilenfenster, um Clientbefehle auszuführen.
   - Fügen Sie dem Grid Daten hinzu:
     - UNIX Linux `./runclient.sh i key1 helloWorld`
     - Windows `runclient.bat i key1 helloWorld`
   - Suchen und zeigen Sie den Wert an:
     - UNIX Linux `./runclient.sh g key1`
     - Windows `runclient.bat g key1`
   - Aktualisieren Sie den Wert:
     - UNIX Linux `./runclient.sh u key1 goodbyeWorld`
     - Windows `runclient.bat u key1 goodbyeWorld`
   - Löschen Sie den Wert:

- UNIX Linux `./runclient.sh d key1`
- Windows `runclient.bat d key1`

6. Verwenden Sie die Tastenkombination <STRG+c>, um den Katalogserviceprozess und die Containerserver in den entsprechenden Fenstern zu stoppen.

ObjectGrid definieren

Das Muster verwendet die Dateien `objectgrid.xml` und `deployment.xml` aus dem Verzeichnis *Installationsstammverzeichnis*`/ObjectGrid/gettingstarted/xml`, um einen Containerserver zu starten. Die Datei `objectgrid.xml` ist die ObjectGrid-XML-Deskriptordatei und die Datei `deployment.xml` die XML-Deskriptordatei für die ObjectGrid-Implementierungsrichtlinien. Beide Dateien zusammen definieren eine verteilte ObjectGrid-Topologie.

ObjectGrid-XML-Deskriptordatei

Eine ObjectGrid-XML-Deskriptordatei wird verwendet, um die Struktur des ObjectGrids definieren, das von der Anwendung verwendet wird. Sie enthält eine Liste mit BackingMap-Konfigurationen. Diese BackingMaps sind der eigentliche Datenspeicher für zwischengespeicherte Daten. Im Folgenden sehen Sie eine Musterdatei `objectgrid.xml`. Die ersten Zeilen der Datei enthalten den erforderlichen Header für jede ObjectGrid-XML-Datei. Diese Musterdatei definiert das Grid "ObjectGrid" mit den BackingMaps "Map1" und "Map2".

```
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

    <objectGrids>
        <objectGrid name="Grid">
            <backingMap name="Map1" />
            <backingMap name="Map2" />
        </objectGrid>
    </objectGrids>

</objectGridConfig>
```

XML-Deskriptordatei für Implementierungsrichtlinien

Eine XML-Deskriptordatei für Implementierungsrichtlinien wird während des Starts an einen ObjectGrid-Containerserver übergeben. Eine Implementierungsrichtlinie muss zusammen mit einer ObjectGrid-XML-Datei verwendet werden und mit der ObjectGrid-XML kompatibel sein, mit der sie verwendet wird. Für jedes objectgridDeployment-Element in der Implementierungsrichtlinie muss ein entsprechendes ObjectGrid-Element in der ObjectGrid-XML vorhanden sein. Die backingMap-Elemente, die im objectgridDeployment-Element definiert werden, müssen mit den backingMap-Elementen in der ObjectGrid-XML konsistent sein. Jedes backingMap-Element darf nur in einem einzigen mapSet-Element referenziert werden.

Die XML-Deskriptordatei für Implementierungsrichtlinien ist für die Verwendung mit der entsprechenden ObjectGrid-XML-Datei `objectgrid.xml` bestimmt. Im folgenden Beispiel enthalten die ersten Zeilen der Datei `deployment.xml` den erforderlichen Header für jede XML-Deskriptordatei für Implementierungsrichtlinien. Die Datei definiert das Element "objectgridDeployment" für das Grid "ObjectGrid", das in der Datei `objectgrid.xml` definiert ist. Beide im Grid "ObjectGrid" definierten BackingMaps, Map1 und Map2, sind im MapSet "mapSet" enthalten, in dem die Attribute "numberOfPartitions", "minSyncReplicas" und "maxSyncReplicas" konfiguriert sind.

```
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy
 ../deploymentPolicy.xsd"
 xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

    <objectgridDeployment objectgridName="Grid">
        <mapSet name="mapSet" numberOfPartitions="13" minSyncReplicas="0"
     maxSyncReplicas="1" >
            <map ref="Map1"/>
            <map ref="Map2"/>
        </mapSet>
    </objectgridDeployment>

</deploymentPolicy>
```

Mit dem Attribut "numberOfPartitions" des Elements "mapSet" wird die Anzahl der Partitionen für das MapSet angegeben. Es ist ein optionales Attribut und hat standardmäßig den Wert 1. Die Anzahl der Partitionen muss der geplanten Grid-Kapazität angemessen sein.

Mit dem Attribut "minSyncReplicas" des MapSets wird die Mindestanzahl synchroner Replikate für jede Partition im MapSet angegeben. Es ist ein optionales Attribut und hat standardmäßig den Wert 0. Es werden erst dann primäre Shards und Replikat-Shards verteilt, wenn die Domäne in der Lage ist, die Mindestanzahl synchroner Replikate zu unterstützen. Für die Unterstützung des minSyncReplicas-Werts benötigen Sie einen Container mehr, als der minSyncReplicas-Wert vorgibt. Wenn die Anzahl synchroner Replikate unter den Wert von minSyncReplicas fällt, werden keine Schreiboperationen für diese Partition mehr zugelassen.

Mit dem Attribut "maxSyncReplicas" des MapSets wird die maximale Anzahl synchroner Replikate für jede Partition im MapSet angegeben. Es ist ein optionales Attribut und hat standardmäßig den Wert 0. Es werden keine weiteren synchronen Replikate für eine Partition verteilt, wenn eine Domäne diese Anzahl synchroner Replikate für diese bestimmte Partition erreicht. Das Hinzufügen von Containern, die dieses ObjectGrid unterstützen, kann zu einer höheren Anzahl synchroner Replikate führen, wenn der maxSyncReplicas-Wert noch nicht erreicht ist. Das Muster setzt maxSyncReplicas auf 1, d. h., die Domäne verteilt maximal ein synchrones Replikat. Wenn Sie mehrere Containerserverinstanzen starten, wird nur ein einziges synchrones Replikat in einer der Containerserverinstanzen verwendet.

ObjectGrid verwenden

Die Datei `Client.java` im Verzeichnis *`Installationsstammverzeichnis`*`/ObjectGrid/gettingstarted/src/`" ist das Clientprogramm, das demonstriert, wie eine Verbindung zum Katalogserver hergestellt, eine ObjectGrid-Instanz abgerufen und die API "ObjectMap" verwendet wird.

Aus der Perspektive einer Clientanwendung kann die Verwendung von WebSphere eXtreme Scale in die folgenden Schritte unterteilt werden:

1. Verbindung zum Katalogservice durch Anfordern einer ClientClusterContext-Instanz herstellen
2. ObjectGrid-Clientinstanz anfordern
3. Session-Instanz abrufen
4. ObjectMap-Instanz abrufen
5. ObjectMap-Methoden verwenden

1. **Verbindung zum Katalogservice durch Anfordern einer ClientClusterContext-Instanz herstellen**

Verwenden Sie zum Herstellen einer Verbindung zum Katalogserver die Methode "connect" der API "ObjectGridManager". Die verwendete Methode "connect" erfordert lediglich einen Katalogserverendpunkt im Format `Hostname:Port`, wie z. B. `localhost:2809`. Wenn die Verbindung zum Katalogserver erfolgreich hergestellt werden kann, gibt die Methode "connect" eine ClientClusterContext-Instanz zurück. Die ClientClusterContext-Instanz ist erforderlich, um das ObjectGrid von der API "ObjectGridManager" abzurufen. Das folgende Code-Snippet veranschaulicht, wie eine Verbindung zu einem Katalogserver hergestellt und eine ClientClusterContext-Instanz angefordert wird.

```
ClientClusterContext ccc = ObjectGridManagerFactory.getObjectGridManager().connect("localhost:2809", null, null);
```

2. **ObjectGrid-Instanz anfordern**

   Zum Anfordern einer ObjectGrid-Instanz verwenden Sie die Methode "getObjectGrid" der API "ObjectGridManager". Die Methode "getObjectGrid" erfordert die ClientClusterContext-Instanz und den Namen der ObjectGrid-Instanz. Die ClientClusterContext-Instanz wird während der Verbindung zum Katalogserver angefordert. Der Name des ObjectGrids ist der Name des `Grids`, das in der Datei `objectgrid.xml` angegeben ist. Das folgende Code-Snippet veranschaulicht, wie eine ObjectGrid-Instanz durch Aufruf der Methode "getObjectGrid" der API "ObjectGridManager" angefordert wird.

   ```
   ObjectGrid grid = ObjectGridManagerFactory.getObjectGridManager().getObjectGrid(ccc, "Grid");
   ```

3. **Session-Instanz abrufen**

   Sie können eine Session-Instanz von der angeforderten ObjectGrid-Instanz abrufen. Eine Session-Instanz ist erforderlich, um die ObjectMap-Instanz abzurufen und die Transaktionsdemarkation durchzuführen. Das folgende Code-Snippet veranschaulicht, wie eine Session-Instanz durch Aufruf der Methode "getSession" der API "ObjectGrid" abgerufen wird.

   ```
   Session sess = grid.getSession();
   ```

4. **ObjectMap-Instanz abrufen**

   Nach dem Abrufen einer Session-Instanz können Sie durch Aufruf der Methode "getMap" der API "Session" eine ObjectMap-Instanz von der Session-Instanz abrufen. Sie müssen den Namen der Map als Parameter an die Methode "getMap" übergeben, um die ObjectMap-Instanz abzurufen. Das folgende Code-Snippet veranschaulicht, wie eine ObjectMap-Instanz durch Aufruf der Methode "getMap" der API "Session" angefordert wird.

   ```
   ObjectMap map1 = sess.getMap("Map1");
   ```

5. **ObjectMap-Methoden verwenden**

   Nach dem Abruf einer ObjectMap-Instanz können Sie die API "ObjectMap" verwenden. Beachten Sie, dass die Schnittstelle "ObjectMap" eine transaktionsorientierte Map ist und eine Transaktionsdemarkation durch die Verwendung der Methoden "begin" und "commit" der API "Session" erfordert. Wenn keine explizite Transaktionsdemarkation in der Anwendung stattfindet, werden die ObjectMap-Operationen über Transaktionen mit automatischer Festschreibung ausgeführt.

   Das folgende Code-Snippet veranschaulicht, wie die API "ObjectMap" mit einer Transaktion mit automatischer Festschreibung verwendet wird.

   ```
   map1.insert(key1, value1);
   ```

   Das folgende Code-Snippet veranschaulicht, wie die API "ObjectMap" mit expliziter Transaktionsdemarkation verwendet wird.

   ```
   sess.begin();
   map1.insert(key1, value1);
   sess.commit();
   ```

Weitere Informationen

Dieses Muster demonstriert, wie Sie einen Katalogserver und einen Containerserver starten und die API "ObjectMap" in einer eigenständigen Umgebung verwenden. Sie können auch die API "EntityManager" verwenden.

In einer Umgebung mit WebSphere Application Server, in der WebSphere eXtreme Scale installiert oder aktiviert ist, ist das gängigste Szenario eine Topologie mit Netzanschluss. in einer Topologie mit Netzanschluss befindet sich der Katalogserver im Deployment-Manager-Prozess von WebSphere Application Server, und jede Instanz von WebSphere Application Server enthält automatisch einen eXtreme-Scale-Server. Java-EE-Anwendungen (Java™ Platform, Enterprise Edition) müssen nur die ObjectGrid-XML-Deskriptordatei und die XML-Deskriptordatei für ObjectGrid-Implementierungsrichtlinien im Verzeichnis `META-INF` jedes Moduls enthalten, und das ObjectGrid ist automatisch verfügbar. Anschließend kann eine Anwendung eine Verbindung zu einem lokal verfügbaren Katalogserver herstellen und eine ObjectGrid-Instanz zur Verwendung abrufen.

Verzeichniskonventionen

In diesem Abschnitt werden zahlreiche Beispiele und die Befehlszeilensyntax beschrieben, in denen spezielle Verzeichnisse wie das *`WXS-Installationsstammverzeichnis`* und *`WXS-Ausgangsverzeichnis`* referenziert werden müssen. Diese Verzeichnisse sind wie folgt definiert.

WXS-Installationsstammverzeichnis
Das Verzeichnis *WXS-Installationsstammverzeichnis* ist das Stammverzeichnis, in dem die Produktdateien von WebSphere eXtreme Scale installiert sind. Dies kann das Verzeichnis sein, in dem die ZIP-Datei mit der Testversion entpackt wurde, oder das Verzeichnis, in dem das vollständige Produkt eXtreme Scale installiert ist.

- Beispiel für die entpackte Testversion:

  ```
  /opt/IBM/WebSphere/eXtremeScale
  ```

- Beispiel für eine Installation von eXtreme Scale in einem eigenständigen Verzeichnis:

  ```
  /opt/IBM/eXtremeScale
  ```

- Beispiel für die Integration von eXtreme Scale mit WebSphere Application Server:

  ```
  /opt/IBM/WebSphere/AppServer
  ```

WXS-Ausgangsverzeichnis
Das Verzeichnis *WXS-Ausgangsverzeichnis* ist das Stammverzeichnis der Produktbibliotheken, Muster und Komponenten von WebSphere eXtreme Scale. Dieses Verzeichnis ist identisch mit dem Verzeichnis *WXS-Installationsstammverzeichnis*, wenn die Testversion entpackt wird. Bei eigenständigen Installationen ist dies das ObjectGrid-Unterverzeichnis im *WXS-Installationsstammverzeichnis*. Bei Installationen, die mit WebSphere Application Server integriert sind, ist dieses Verzeichnis das Verzeichnis `optionalLibraries/ObjectGrid` im *WXS-Installationsstammverzeichnis*.

- Beispiel für die entpackte Testversion:

  ```
  /opt/IBM/WebSphere/eXtremeScale
  ```

- Beispiel für eine Installation von eXtreme Scale in einem eigenständigen Verzeichnis:

  ```
  /opt/IBM/eXtremeScale/ObjectGrid
  ```

- Beispiel für die Integration von eXtreme Scale mit WebSphere Application Server:

  ```
  /opt/IBM/WebSphere/AppServer/optionalLibraries/ObjectGrid
  ```

WAS-Stammverzeichnis
Das Verzeichnis *WAS-Stammverzeichnis* ist das Stammverzeichnis einer Installation von WebSphere Application Server:

```
/opt/IBM/WebSphere/AppServer
```

Ausgangsverzeichnis_des_REST-Service
Das Verzeichnis *Ausgangsverzeichnis_des_REST-Service* ist das Verzeichnis, in dem sich die Bibliotheken und Muster des REST-Datenservice von eXtreme Scale befinden. Das Verzeichnis hat den Namen *restservice* und ist ein Unterverzeichnis des Verzeichnisses *WXS-Ausgangsverzeichnis*.

- Beispiel für eigenständige Implementierungen:

  ```
  /opt/IBM/WebSphere/eXtremeScale/ObjectGrid/restservice
  ```

- Beispiel für integrierte Implementierungen mit WebSphere Application Server:

  ```
  /opt/IBM/WebSphere/AppServer/optionalLibraries/ObjectGrid/restservice
  ```

Tomcat-Stammverzeichnis
Das *Tomcat-Stammverzeichnis* ist das Stammverzeichnis der Apache-Tomcat-Installation.

```
/opt/tomcat5.5
```

WASCE-Stammverzeichnis
Das *WASCE-Stammverzeichnis* ist das Stammverzeichnis der Installation von WebSphere Application Server Community Edition.

```
/opt/IBM/WebSphere/AppServerCE
```

Java-Ausgangsverzeichnis
Das *Java-Ausgangsverzeichnis* ist das Stammverzeichnis der Installation von Java Runtime Environment (JRE).

```
/opt/IBM/WebSphere/eXtremeScale/java
```

Kapitel 2. Kapazitätsplanung

Wenn Sie eine anfängliche Dateigruppengröße und eine geplante Dateigruppengröße haben, können Sie die für die Ausführung von WebSphere eXtreme Scale erforderliche Kapazität planen. Eine solche Planung hilft Ihnen nicht nur bei einer effizienten Implementierung von eXtreme Scale in Bezug auf künftige Änderungen, sondern ermöglicht Ihnen auch, die Elastizität von eXtreme Scale zu maximieren, was mit einem anderen Szenario, wie beispielsweise mit einer speicherinternen Datenbank oder einem anderen Typ von Datenbank, nicht möglich ist.

Übersicht über Skalierbarkeitskonzepte

Skalierbarkeit ermöglicht in einer Implementierung von WebSphere eXtreme Scale je nach ausgewählter Konfiguration die Verteilung von Daten auf eine Gruppe von Servern (Containern).

Speicher dimensionieren und Partitionsanzahl berechnen

Sie können die für Ihre spezielle Konfiguration benötigte Speicherkapazität und Partitionsanzahl berechnen.

WebSphere eXtreme Scale speichert Daten im Adressraum von Java Virtual Machines (JVM). Jede JVM stellt Prozessorplatz für die Bearbeitung von Erstellungs-, Abruf-, Aktualisierungs- und Löschaufrufen für Daten bereit, die in der JVM gespeichert sind. Außerdem stellt jede JVM Speicherplatz für Dateneinträge und Replikate bereit. Java-Objekte variieren in ihrer Größe, und deshalb müssen Sie Messungen durchführen, um die benötigte Speicherkapazität zu schätzen.

Zur Dimensionierung des benötigten Speichers laden Sie Ihre Anwendungsdaten in eine einzige JVM. Wenn die Heap-Speicherbelegung einen Wert von 60 % erreicht, notieren Sie die Anzahl der verwendeten Objekte. Diese Zahl ist die empfohlene maximale Objektanzahl für jede Ihrer JVMs. Für eine möglichst genaue Dimensionierung sollten Sie realistische Daten verwenden und alle definierten Indizes einbeziehen, weil Indizes auch Speicher belegen. Die beste Methode für die Speicherdimensionierung ist die Durchführung einer ausführlichen Garbage-Collection (mit verbosegc), da diese Ausgabe Ihnen die Zahlen nach der Garbage-Collection liefert. Sie können die Heap-Speicherbelegung jederzeit über MBeans oder über das Programm abfragen, aber diese Abfragen liefern Ihnen nur eine aktuelle Momentaufnahme des Heap-Speichers, die nicht erfassten Garbage (fehlerhafte Daten) enthalten kann. Deshalb ist die Verwendung dieser Methode keine genaue Indikation für den belegten Speicher.

Konfiguration vertikal skalieren

Anzahl der Shards pro Partition (numShardsPerPartition)

Zum Berechnen der Shard-Anzahl pro Partition bzw. des Werts von "numShardsPerPartition" addieren Sie 1 für das primäre Shard und die Gesamtanzahl der gewünschten Replikat-Shards.

```
numShardsPerPartition = 1 + Gesamtanzahl_der_Replikate
```

Anzahl der JVMs (minNumJVMs)

Für die vertikale Skalierung der Konfiguration müssen Sie zuerst die maximale Anzahl an Objekten festlegen, die insgesamt gespeichert werden müssen. Verwenden Sie die folgende Formel, um die Anzahl der benötigten JVMs zu bestimmen:

```
minNumJVMS=(numShardsPerPartition * numObjs) / numObjsPerJVM
```

Runden Sie diesen Wert auf die nächst höhere ganze Zahl auf.

Anzahl der Shards (numShards)

Bei der endgültigen Größe sollten 10 Shards für jede JVM verwendet werden. Wie zuvor beschrieben, hat jede JVM ein primäres Shard und (N-1) Replikat-Shards bzw. in diesem Fall 9 Replikate. Da Sie bereits eine Zahl für die Java Virtual Machines haben, in denen die Daten gespeichert werden, können Sie die JVM-Zahl mit 10 multiplizieren, um die Anzahl der Shards zu bestimmen:

```
numShards = minNumJVMs * 10 Shards/JVM
```

Anzahl der Partitionen

Wenn eine Partition ein einziges primäres Shard und ein einziges Replikat-Shard hat, hat die Partition zwei Shards (das primäre Shard und das Replikat-Shard). Die Anzahl der Partitionen entspricht der Shard-Anzahl, geteilt durch 2, aufgerundet auf die nächst höhere Primzahl. Wenn die Partition ein primäres Shard und zwei Replikat-Shards hat, entspricht die Anzahl der Partitionen der Shard-Anzahl, geteilt durch 3, aufgerundet auf die nächst höhere Primzahl.

```
numPartitions = numShards / numShardsPerPartition
```

Skalierungsbeispiel

In diesem Beispiel wird von einer anfänglichen Eintragsanzahl von 250 Millionen ausgegangen. Jedes Jahr nimmt die Anzahl der Einträge um etwa 14 % zu. Nach sieben Jahren sind insgesamt 500 Millionen Einträge vorhanden. Deshalb müssen Sie Ihre Kapazität entsprechend planen. Für die hohe Verfügbarkeit wird ein einziges Replikat benötigt. Mit einem Replikat verdoppelt sich die Anzahl der Einträge, d. h. auf eine 1 Milliarde Einträge. Zu Testzwecken können 2 Millionen Einträge in jeder JVM gespeichert werden. Laut den Berechnungen wird dann in diesem Szenario die folgende Konfiguration benötigt:

- 500 JVMs zum Speichern der endgültigen Anzahl an Einträgen
- 5000 Shards (500 Java Virtual Machines mal 10)
- 2500 Partitionen bzw. 2503, da dies die nächst höhere Primzahl ist (5000 Shards, geteilt durch zwei für primäre Shards und Replikat-Shards)

Anfangskonfiguration

Basierend auf den folgenden Berechnungen, beginnen Sie mit 250 Java Virtual Machines und steigern sich dann in einem Zeitraum von fünf Jahren auf 500 Java Virtual Machines. Damit können Sie das inkrementelle Wachstum verwalten, bis Sie die endgültige Anzahl an Einträgen erreichen.

In dieser Konfiguration werden ungefähr 200.000 Einträge pro Partition (500 Millionen Einträge, geteilt durch 2503 Partitionen) gespeichert. Sie sollten den Parameter "numberOfBuckets" in der Map, die die Einträge enthält, auf die nächst höhere Primzahl setzen (in diesem Beispiel 70887), die das Verhältnis bei ungefähr 3 hält.

Erreichen der maximalen Anzahl an Java Virtual Machines

Wenn Sie die maximale Anzahl von 500 Java Virtual Machines erreichen, können Sie Ihr Grid trotzdem weiter vergrößern. Da die Anzahl der Java Virtual Machines über 500 steigt, fällt die Shard-Anzahl für jede JVM unter 10, was unter der empfohlenen Zahl liegt. Die Shards werden größer, was zu Problemen führen kann. Sie müssen den Dimensionierungsprozess unter Berücksichtigung des künftigen Wachstums wiederholen und die Partitionsanzahl zurücksetzen. Dieses Verfahren erfordert einen vollständigen Neustart bzw. Außerbetriebnahme des Grids.

Anzahl der Server

Achtung: Verwenden Sie unter keinen Umständen Paging auf einem Server.

Die Speicherbelegung einer einzigen JVM ist höher als die Größe des Heap-Speichers. Bei einem Heap-Speicher mit einer Größe von 1 GB für eine JVM werden tatsächlich 1,4 GB Realspeicher belegt. Bestimmen Sie den verfügbaren freien Arbeitsspeicher des Servers. Teilen Sie die Arbeitsspeicherkapazität durch den Speicher pro JVM, um die maximale Anzahl an Java Virtual Machines auf dem Server zu erhalten.

CPU-Dimensionierung pro Partition für Transaktionen

Obwohl die Hauptfunktionalität von eXtreme Scale die elastische Skalierung ist, ist es auch wichtig, die optimale Anzahl an CPUs zu dimensionieren und anzupassen, um eine vertikale Skalierung zu erreichen.

Die Prozessorkosten setzen sich wie folgt zusammen:

- Kosten für die Verarbeitung von Erstellungs-, Abruf-, Aktualisierungs- und Löschoperationen von Clients,
- Kosten für die Replikation von Daten anderer Java Virtual Machines,
- Kosten für die Inaktivierung,
- Kosten für die Bereinigungsrichtlinie,
- Kosten für die Garbage-Collection,
- Kosten für die Anwendungslogik.

Java Virtual Machines pro Server

Verwenden Sie zwei Server, und starten Sie die maximale Anzahl an JVMs pro Server. Verwenden Sie die im vorherigen Abschnitt berechnete Partitionsanzahl. Laden Sie vorab nur so viele Daten in die Java Virtual Machines, wie auf diese beiden Computer passen. Verwenden Sie einen gesonderten Server als Client. Führen Sie eine realistische Transaktionssimulation für das Grid der beiden Server durch.

Versuchen Sie die Prozessorauslastung zu sättigen, um das Ausgangsniveau zu berechnen. Sollte dies nicht möglich sein, ist das Netz wahrscheinlich gesättigt. Wenn das Netz gesättigt ist, fügen Sie weitere Netzkarten hinzu, und verteilen Sie die Java Virtual Machines auf die verfügbaren Netzkarten.

Führen Sie die Computer mit einer Prozessorauslastung von 60 % aus, und messen Sie die Raten für die Erstellungs- (create), Abruf- (retrieve), Aktualisierungs- (update) und Löschtransaktionen (delete). Diese Messung liefert Ihnen den Durchsatz auf den beiden Servern. Diese Zahl verdoppelt sich bei vier Servern und verdoppelt sich dann nochmal bei 8 Servern usw. Bei dieser Skalierung wird davon ausgegangen, dass die Netzkapazität und die Clientkapazität ebenfalls skalierbar sind.

Die Antwortzeiten von eXtreme Scale sollten mit zunehmender Anzahl an Servern somit stabil bleiben. Der Transaktionsdurchsatz sollte linear steigen, wenn dem Grid Computer hinzugefügt werden.

CPU-Dimensionierung für parallele Transaktionen

Einzelpartitionstransaktionen haben einen Durchsatz, der linear zum Wachstum des Grids steigt. Parallele Transaktionen unterscheiden sich von Einzelpartitionstransaktionen, weil sie einen Teil der Server (oder auch alle Server) betreffen.

Wenn eine Transaktion alle Server betrifft, ist der Durchsatz auf den Durchsatz des Clients, der die Transaktion einleitet, bzw. auf den Durchsatz des langsamsten betroffenen Servers beschränkt. In größeren Grids werden die Daten breiter verteilt. Diese Grids stellen mehr Prozessorplatz, Speicherplatz, Netzkapazität usw. bereit. Der Client muss jedoch auf die Antwort des langsamsten Servers warten, und der Client muss die Ergebnisse der Transaktion konsumieren.

Wenn eine Transaktion einen Teil der Server betrifft, erhalten M von N Servern eine Anforderung. Der Durchsatz ist dann N/M-Mal so hoch wie der Durchsatz des langsamsten Servers. Wenn Sie beispielsweise 20 Server und eine Transaktion haben, die 5 Server betrifft, ist der Durchsatz 4 Mal so hoch wie der Durchsatz des langsamsten Servers im Grid.

Nach Abschluss einer parallelen Transaktion werden die Ergebnisse an den Client-Thread gesendet, der die Transaktion gestartet hat. Dieser Client muss daraufhin die Ergebnisse in Einzel-Threads zusammenfassen. Die Dauer dieser Zusammenfassung (oder Aggregation) erhöht sich mit zunehmender Anzahl der von der Transaktion betroffenen Server. Diese Dauer hängt jedoch von der Anwendung ab, da es möglich ist, dass jeder Server bei zunehmender Größe des Grids ein kleineres Ergebnis zurückgibt.

Gewöhnlich betreffen parallele Transaktionen alle Server im Grid, weil Partitionen gleichmäßig auf das Grid verteilt werden. In diesem Fall ist der Durchsatz, wie im ersten Fall beschrieben, beschränkt.

Zusammenfassung

Für diese Dimensionierung stehen Ihnen drei Messwerte zur Verfügung:

- Anzahl der Partitionen,
- Anzahl der Server, die für den erforderlichen Speicherbedarf benötigt werden,
- Anzahl der Server, die für den erforderlichen Durchsatz benötigt werden.

Wenn Sie 10 Server für den Speicherbedarf benötigen, aber aufgrund der Prozessorsättigung nur 50 % des erforderlichen Durchsatzes erzielen, benötigen Sie doppelt so viele Server wie vorhanden.

Die höchste Stabilität erzielen Sie, wenn Sie Ihre Server mit einer Prozessorauslastung von 60 % und Ihre JVMs mit einer Heap-Speicherauslastung von 60 % betreiben. So können Lastspitzen die Prozessorauslastung auf 80–90 % hochtreiben. Ein dauerhafter Betrieb der Server mit diesen Ständen oder höher sollte aber vermieden werden.

Kapazitätsplanung und hohe Verfügbarkeit (dynamisches Caching)

Die Anwendungsprogrammierschnittstelle (API, Application Programming Interface) für dynamischen Cache steht Java-EE-Anwendungen zur Verfügung, die in WebSphere Application Server implementiert sind. Der dynamische Cache kann genutzt werden, um Geschäftsdaten und generierte HTML zwischenzuspeichern oder um die zwischengespeicherten Daten in der Zelle über den Datenreplikationsservice (DRS) zu synchronisieren.

Übersicht

Alle dynamischen Cacheinstanzen, die mit dem dynamischen Cacheprovider von WebSphere eXtreme Scale erstellt werden, sind standardmäßig hoch verfügbar. Die Stufe und die Speicherkosten der hohen Verfügbarkeit sind von der verwendeten Topologie abhängig.

Wenn Sie die integrierte Topologie verwenden, ist die Cachegröße auf den freien Speicher in einem einzelnen Serverprozess beschränkt, und in jedem Serverprozess wird eine vollständige Kopie des Caches gespeichert. Solange auch nur ein einziger Serverprozess aktiv ist, bleibt auch der Cache aktiv. Die Cachedaten gehen nur dann verloren, wenn alle Server, die auf den Cache zugreifen, beendet werden.

Beim Caching mit der integrierten partitionierten Topologie ist die Cachegröße auf den summierten freien Speicher aller Serverprozesse beschränkt. Standardmäßig verwendet der dynamische Cacheprovider von eXtreme Scale ein Replikat für jedes primäre Shard, d. h., jedes einzelne zwischengespeicherte Datenelement wird zweimal gespeichert.

Verwenden Sie die folgende Formel A, um die Kapazität eines integrierten partitionierten Caches zu bestimmen:

Formel A

F * C / (1 + R) = M

Für diese Formel gilt Folgendes:

- F = Freier Speicher pro Containerprozess
- C = Anzahl der Container
- R = Anzahl der Replikate
- M = Gesamtgröße des Caches

Für ein Grid von WebSphere Network Deployment mit 256 MB verfügbarem Speicher in jedem Prozess und 4 Serverprozessen insgesamt können in einer Cacheinstanz über alle diese Server verteilt 512 Megabyte Daten gespeichert werden. In diesem Modus kann der Cache einen Serverabsturz ohne Datenverlust "überleben". Es könnten sogar nacheinander zwei Server beendet werden, ohne dass irgendwelche Daten verloren gehen. Für das vorherige Beispiel lautet die Formel deshalb wie folgt:

256 MB * 4 Container/ (1 primäres Shard + 1 Replikat) = 512 MB

Caches, die die ferne Topologie verwenden, haben ähnliche Größenmerkmale wie Caches, die die integrierte partitionierte Topologie verwenden, sind jedoch auf den summierten verfügbaren Speicher aller Containerprozesse von eXtreme Scale beschränkt.

In fernen Topologien kann die Anzahl der Replikate erhöht werden, um eine höhere Stufe der Verfügbarkeit zu erreichen, wodurch sich jedoch der Speicheraufwand erhöht. In den meisten dynamischen Cacheanwendungen ist dies in der Regel nicht erforderlich, aber Sie können die Datei "dynacache-remote-deployment.xml" bearbeiten, um die Anzahl der Replikate zu erhöhen.

Verwenden Sie die folgenden Formeln (B und C), um zu berechnen, welche Auswirkungen das Hinzufügen weiterer Replikate auf die hohe Verfügbarkeit des Caches hat.

Formel B

N = Minimum(T -1, R)

Für diese Formel gilt Folgendes:

- N = Anzahl der Prozesse, die gleichzeitig abstürzen
- T = Gesamtanzahl der Container
- R = Gesamtanzahl der Replikate

Formel C

Obere Grenze(T/ (1+N)) = m

Für diese Formel gilt Folgendes:

- T = Gesamtanzahl der Container
- N = Gesamtanzahl der Replikate
- m = Erforderliche Mindestanzahl an Containern für die Unterstützung der Cachedaten

Informationen zur Leistungsoptimierung mit dem dynamischen Cacheprovider finden Sie im Abschnitt Dynamischen Cacheprovider optimieren.

Cachegröße festlegen

Bevor eine Anwendung, die den dynamischen Cacheprovider von WebSphere eXtreme Scale verwendet, implementiert werden kann, müssen die allgemeinen Principals, die im vorherigen Abschnitt beschrieben wurden, mit den Umgebungsdaten für die Produktionssysteme kombiniert werden. Der erste zu ermittelnde Wert ist die Gesamtanzahl der Containerprozesse und der verfügbare Hauptspeicher jedes einzelnen Prozesses zum Speichern der Cachedaten. Wenn Sie die integrierte Topologie verwenden, befinden sich die Cachecontainer in den Prozessen von WebSphere Application Server. d. h., es gibt einen Container pro Server, der den Cache verwendet. Die Bestimmung der Speicherkosten der Anwendung ohne aktiviertes Caching und ohne WebSphere Application Server ist die beste Methode zu ermitteln, wie viel Speicher im Prozess verfügbar ist. Zur Ermittlung dieses Werts bietet sich die Analyse der Daten einer ausführlichen Garbage-Collection an. Wenn Sie die ferne Topologie verwenden, können diese Informationen anhand der Ausgabe der ausführlichen Garbage-Collection für einen neu gestarteten eigenständigen Container ermittelt werden, der noch nicht mit Cachedaten gefüllt wurde. Ein letzter Aspekt, der bei der Ermittlung des für Cachedaten verfügbaren Speichers pro Prozess berücksichtigt werden muss, ist die Reservierung einer gewissen Menge des Heap-Speichers für die Garbage-Collection. Die Summe aus Containerkosten

(Umgebung mit WebSphere Application Server oder eigenständige Umgebung) und reservierter Größe für den Cache sollte nicht mehr als 70 % der Gesamtgröße des Heap-Speichers betragen.

Nachdem Sie diese Informationen zusammengestellt haben, können Sie die Werte in die zuvor beschriebene Formel A eintragen, um die maximale Größe für den partitionierten Cache zu bestimmen. Sobald die maximale Größe bekannt ist, müssen Sie im nächsten Schritt die Gesamtanzahl der Cacheeinträge bestimmen, die unterstützt werden können. Hierfür muss zunächst die durchschnittliche Größe pro Cacheeintrag bestimmt werden. Eine einfache Methode zur Bestimmung dieses Werts ist, 10 % auf die Größe des Kundenobjekts aufzuschlagen. Ausführlichere Informationen zur Festlegung der Größe von Cacheeinträgen bei der Verwendung des dynamischen Caches finden Sie in der Veröffentlichung "Tuning guide for dynamic cache and data replication service".

Wenn die Komprimierung aktiviert ist, wirkt diese sich auf die Größe des Kundenobjekts aus, aber nicht auf die Kosten des Caching-Systems. Verwenden Sie die folgende Formel, um die Größe eines zwischengespeicherten Objekts zu bestimmen, wenn Sie mit Komprimierung arbeiten:

S = O * C + O * 0.10

Für diese Formel gilt Folgendes:

- S = Durchschnittliche Größe eines zwischengespeicherten Objekts
- O = Durchschnittliche Größe eines nicht komprimierten Kundenobjekts
- C = Komprimierungsfaktor als Bruchzahl

Komprimierung: Ein Komprimierungsfaktor von 2 zu 1 ist 1/2 = 0,50. Je kleiner der Faktor ist, desto besser. Wenn das zu speichernde Objekt ein normales POJO mit überwiegend primitiven Datentypen ist, können Sie von einem Komprimierungsfaktor von 0,60 bis 0,70 ausgehen. Wenn das zwischengespeicherte Objekte ein Servlet, eine JSP-Datei oder ein Web-Service-Objekt ist, ist die optimale Methode für die Bestimmung des Komprimierungsfaktors, ein repräsentatives Beispielobjekt mit einem Komprimierungsdienstprogramm zu komprimieren. Sollte dies nicht möglich sein, können Sie im Allgemeinen von einem Komprimierungsfaktor von 0,2 bis 0,35 für diesen Typ von Daten ausgehen.

Anschließend verwenden Sie diese Informationen, um die Gesamtanzahl der Cacheeinträge zu bestimmen, die unterstützt werden können. Verwenden Sie die folgende Formel D:

Formel D

T = S / A

Für diese Formel gilt Folgendes:

- T = Gesamtanzahl der Cacheeinträge
- S = Verfügbare Gesamtkapazität für Cachedaten laut Berechnung mit Formel A
- A = Durchschnittliche Größe eines Cacheeintrags

Abschließend müssen Sie die Cachegröße in der dynamischen Cacheinstanz festlegen, damit dieser Grenzwert wirksam wird. Der dynamische Cacheprovider von WebSphere eXtreme Scale unterscheidet sich in dieser Beziehung vom dynami-

schen Standardcacheprovider. Verwenden Sie die folgende Formel E, um den Wert zu bestimmen, der für die Cachegröße in der dynamischen Cacheinstanz festzulegen ist:

Formel E

Cs = Ts / Np

Für diese Formel gilt Folgendes:

- Ts = Gesamtzielgröße für den Cache
- Cs = In der dynamischen Cacheinstanz festzulegende Cachegröße
- Np = Anzahl der Partitionen (der Standardwert ist 47)

Setzen Sie die Größe der dynamischen Cacheinstanz in allen Servern, die die Cacheinstanz gemeinsam nutzen, auf den mit der Formel E berechneten Wert.

Kapitel 3. WebSphere eXtreme Scale installieren und implementieren

WebSphere eXtreme Scale ist ein speicherinternes Daten-Grid, das Sie verwenden können, um Anwendungsdaten und Geschäftslogik in mehreren Servern zu partitionieren, zu replizieren und zu verwalten. Nachdem Sie den Zweck und die Anforderungen Ihrer Implementierung bestimmt haben, installieren Sie eXtreme Scale auf Ihrem System.

Vorbereitende Schritte

- Stellen Sie fest, wie WebSphere eXtreme Scale in Ihre aktuelle Topologie passt. Weitere Informationen finden Sie in der Übersicht über die Architektur und Topologie von WebSphere eXtreme Scale in der *Produktübersicht*.
- Vergewissern Sie sich, dass Ihre Umgebung die Voraussetzungen für die Installation von eXtreme Scale erfüllt. Weitere Informationen finden Sie im Abschnitt „Hardware- und Softwarevoraussetzungen” auf Seite 65.

Informationen zu diesem Vorgang

7.1+ **Zwei Installationstypen**

- Vollständige Installation von WebSphere eXtreme Scale: Sie können diese Installation verwenden, um den Server und/oder den Client zu installieren.
- Installation von WebSphere eXtreme Scale Client: Sie können diese Installation verwenden, um den Client auf bestimmten Plattformen zu installieren.

Unterstützte Umgebungen

Sie müssen eXtreme Scale nicht unter einer bestimmten Version des Betriebssystems installieren und implementieren. Jede J2SE- (Java Platform, Standard Edition) und jede JEE-Installation (Java Platform, Enterprise Edition) setzt andere Betriebssystemversionen oder -Fixes voraus.

Sie können das Produkt in JEE- und J2SE-Umgebungen installieren. Außerdem können Sie die Clientkomponente direkt, ohne Integration mit WebSphere Application Server, mit JEE-Anwendungen bündeln. WebSphere eXtreme Scale unterstützt Java Runtime Environment (JRE) Version 1.4.2 und höher sowie WebSphere Application Server Version 6.0.2 und höher.

Vorgehensweise

- Installieren Sie die eigenständige Version von WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client.

  Sie können die eigenständige Version von eXtreme Scale in einer Umgebung installieren, die weder WebSphere Application Server noch WebSphere Application Server Network Deployment enthält. Bei der eigenständigen Option definieren Sie eine neue Installationsposition, an der Sie den eXtreme-Scale-Server installieren möchten.

Achtung: Sie können ein Profil ohne Root-Rechte (Administratorrechte) auch für WebSphere eXtreme Scale in einer eigenständigen Umgebung verwenden. Eine eigenständige Umgebung ist eine Umgebung, in der WebSphere Application Server nicht verwendet wird. Wenn Sie ein Profil ohne Root-Rechte verwenden möchten, müssen Sie den Eigner des Verzeichnisses `ObjectGrid` in das Profil ohne Root-Rechte ändern. Anschließend können Sie sich mit diesem Profil ohne Root-Rechte anmelden und wie mit einem Profil mit Root-Rechten (Administratorrechten) mit eXtreme Scale arbeiten.

- Integrieren Sie das Produkt mit WebSphere Application Server oder WebSphere Application Server Network Deployment.

  Sie können eXtreme Scale in einer vorhandenen Installation von WebSphere Application Server oder WebSphere Application Server Network Deployment installieren und integrieren. Bei der vollständigen Installation können Sie den eXtreme-Scale-Client und den eXtreme-Scale-Server installieren, oder Sie können auch nur den Client installieren.
- Erstellen und erweitern Sie Profile.

  Erstellen und erweitern Sie Profile für die Verwendung der Features von eXtreme Scale. Wenn Sie WebSphere Application Server Version 6.1 oder Version 7.0 ausführen, können Sie das PMT-Plug-in (Profile Management Tool) oder den Befehl "manageprofiles" verwenden. Wenn Sie WebSphere Application Server Version 6.0.2 ausführen, müssen Sie zum Erstellen und Erweitern von Profilen den Befehl "wasprofile" verwenden.
- Wenden Sie Wartungspakete an.

  Verwenden Sie IBM® Update Installer Version 7.0.0.4 oder höher, um Wartungspakete auf Ihre Umgebung anzuwenden.

Migration auf WebSphere eXtreme Scale Version 7.1

Mit dem Installationsprogramm von WebSphere eXtreme Scale ist ein Upgrade oder eine Änderung einer vorherigen Installation nicht möglich. Sie müssen die vorherige Version deinstallieren, bevor Sie die neue Version installieren. Die Konfigurationsdateien müssen nicht migriert werden, weil sie abwärtskompatibel sind. Wenn Sie jedoch die mit dem Produkt gelieferten Scriptdateien geändert haben, müssen Sie diese Änderungen erneut auf die aktualisierten Scriptdateien anwenden.

Vorbereitende Schritte

Vergewissern Sie sich, dass Ihre Systeme die Mindestvoraussetzungen für die Produktversionen erfüllen, die Sie migrieren und installieren möchten. Weitere Informationen finden Sie unter „Hardware- und Softwarevoraussetzungen” auf Seite 65.

Informationen zu diesem Vorgang

Führen Sie alle geänderten Produktscriptdateien mit den neuen Produktscriptdateien im Verzeichnis `/bin` zusammen, um Ihre Änderungen beizubehalten.

Tipp: Wenn Sie die mit dem Produkt installierten Scriptdateien nicht geändert haben, müssen Sie die folgenden Migrationsschritte nicht ausführen. Stattdessen können Sie das Upgrade auf Version 7.1 durchführen, indem Sie die vorherige Version deinstallieren und die neue Version in demselben Verzeichnis installieren.

Vorgehensweise

1. Stoppen Sie alle Prozesse, die eXtreme Scale verwenden.

- Lesen Sie den Abschnitt Eigenständige Server stoppen, um alle Prozesse zu stoppen, die in der eigenständigen Umgebung von eXtreme Scale ausgeführt werden.
- Informationen zum Stoppen aller Prozesse, die in Ihrer Umgebung von WebSphere Application Server oder WebSphere Application Server Network Deployment ausgeführt werden, finden Sie unter "Befehlszeilendienstprogramme".

2. Speichern Sie alle geänderten Scripts aus Ihrem aktuellen Installationsverzeichnis in einem temporären Verzeichnis.
3. Deinstallieren Sie das Produkt.
4. Installieren Sie eXtreme Scale Version 7.1. Weitere Informationen finden Sie in Kapitel 3, „WebSphere eXtreme Scale installieren und implementieren”, auf Seite 17.
5. Fügen Sie Ihre Änderungen aus den Dateien im temporären Verzeichnis in die neuen Produktscriptdateien im Verzeichnis `/bin` ein.
6. Starten Sie alle Prozesse von eXtreme Scale, um mit der Verwendung des Produkts zu beginnen. Weitere Informationen finden Sie im Abschnitt „Verwaltungsterminologie” auf Seite 323.

WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client standalone installieren

Sie können WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client standalone (eigenständig) in einer Umgebung installieren, in der WebSphere Application Server bzw. WebSphere Application Server Network Deployment nicht vorhanden ist.

Vorbereitende Schritte

- Stellen Sie sicher, dass das Zielinstallationsverzeichnis leer bzw. nicht vorhanden ist.

  Wichtig: Wenn eine frühere Version von WebSphere eXtreme Scale oder die Komponente "ObjectGrid" in dem Verzeichnis vorhanden ist, das Sie für die Installation von Version 7.1 angeben, wird das Produkt nicht installiert. Sie können beispielsweise bereits einen Ordner `<WXS-Installationsstammverzeichnis>/ObjectGrid` haben. In diesem Fall können Sie ein anderes Installationsverzeichnis angeben oder die Installation abbrechen. Anschließend deinstallieren Sie die frühere Installation und führen dann den Assistenten erneut aus.

- **7.1+** Eine IBM Runtime Environment wird im Rahmen der eigenständigen Installation im Ordner *<Ausgangsverzeichnis_der_WXS-Installation>*`/java` installiert.

Informationen zu diesem Vorgang

Wenn Sie das Produkt als eigenständiges Produkt installieren, installieren Sie den Client und den Server von WebSphere eXtreme Scale unabhängig voneinander. Mit der Installation von WebSphere eXtreme Scale Client im eigenständigen Modus installieren Sie einen Client für den Zugriff auf die Daten in Ihren Daten-Grids. Server- und Clientprozesse greifen deshalb auf alle erforderlichen Ressourcen lokal zu. Sie können WebSphere eXtreme Scale auch mit Scripts und JAR-Dateien (Java-Archiv) in vorhandene J2SE-Anwendungen (Java Platform, Standard Edition) integrieren.

Tabelle 1. Laufzeitdateien für eine vollständige Installation von WebSphere eXtreme Scale. WebSphere eXtreme Scale stützt sich auf ObjectGrid-Prozesse und zugehörige APIs. In der folgenden Tabelle sind die JAR-Dateien aufgelistet, die in der Installation enthalten sind.

Dateiname	Umgebung	Installations-position	Beschreibung
wxsdynacache.jar	Client und Server	dynacache/lib	Die Datei wxsdynacache.jar enthält die erforderlichen Klassen für den dynamischen Cacheprovider. Die Datei wird automatisch in die Serverlaufzeitumgebung eingeschlossen, wenn Sie die bereitgestellten Scripts verwenden.
wxshyperic.jar	Dienstprogramm	hyperic/lib	Das Erkennungs-Plug-in von WebSphere eXtreme Scale für den SpringSource-Hyperic-Überwachungsagenten
objectgrid.jar	Lokal, Client und Server	lib	Die Datei objectgrid.jar wird von der Laufzeitumgebung des Servers von J2SE Version 1.4.2 und höher verwendet. Die Datei wird automatisch in die Serverlaufzeitumgebung eingeschlossen, wenn Sie die bereitgestellten Scripts verwenden.
ogagent.jar	Lokal, Client und Server	lib	Die Datei ogagent.jar enthält die Laufzeitklassen, die für die Ausführung des Java-Instrumentierungsagenten erforderlich sind, der mit der API "EntityManager" verwendet wird.
ogclient.jar	Lokal und Client	lib	Die Datei ogclient.jar enthält nur die lokale Laufzeitumgebung und die Clientlaufzeitumgebung. Sie können diese Datei mit J2SE Version 1.4.2 und höher verwenden.
ogspring.jar	Lokal, Client und Server	lib	Die Datei ogspring.jar lt enthält Unterstützungsklassen für die Integration des SpringSource-Spring-Frameworks.
wsogclient.jar	Lokal und Client	lib	Die Datei wsogclient.jar wird installiert, wenn Sie eine Umgebung verwenden, die WebSphere Application Server Version 6.0.2 und höher verwendet. Diese Datei enthält nur die lokale Laufzeitumgebung und die Clientlaufzeitumgebung.
wxssizeagent.jar	Lokal, Client und Server	lib	Die Datei wxssizeagent.jar wird verwendet, um genauere Größeninformationen zu den Cacheeinträgen bereitzustellen, wenn Java Runtime Environment (JRE) Version 1.5 oder höher verwendet wird.
ibmcfw.jar ibmext.jar ibmorb.jar ibmorbapi.jar	Client und Server	lib/endorsed	Diese Gruppe von Dateien enthält die ORB-Laufzeitumgebung, die für die Ausführung von Anwendungen in Java-SE-Prozessen verwendet wird.
restservice.ear	Client	restservice/lib	Die Datei restservice.ear enthält das Unternehmensarchiv der REST-Datenserviceanwendung von eXtreme Scale für Umgebungen von WebSphere Application Server.
restservice.war	Client	restservice/lib	Die Datei restservice.war enthält das Webarchiv des REST-Datenservice von eXtreme Scale für Anwendungsserver eines anderen Anbieters.
xsadmin.jar	Dienstprogramm	samples	Die Datei xsadmin.jar enthält das Verwaltungsdienstprogramm für die Muster von eXtreme Scale.
sessionobjectgrid.jar	Client und Server	session/lib	Die Datei sessionobjectgrid.jar enthält die Laufzeitumgebung von eXtreme Scale für das HTTP-Sitzungsmanagement.
splicerlistener.jar	Dienstprogramm	session/lib	Die Datei splicerlistener.jar enthält das Dienstprogramm "splicer" für den HTTP-Sitzungs-Listener von eXtreme Scale Version 7.1.
xsgbean.jar	Server	wasce/lib	Die Datei xsgbean.jar enthält die GBean für die Integration von Servern von eXtreme Scale in Anwendungsservern von WebSphere Application Server Community Edition.

Tabelle 1. Laufzeitdateien für eine vollständige Installation von WebSphere eXtreme Scale (Forts.). WebSphere eXtreme Scale stützt sich auf ObjectGrid-Prozesse und zugehörige APIs. In der folgenden Tabelle sind die JAR-Dateien aufgelistet, die in der Installation enthalten sind.

Dateiname	Umgebung	Installations-position	Beschreibung
splicer.jar	Dienstprogramm	legacy/session/lib	Das Dienstprogramm "splicer" für den HTTP-Sitzungsmanagerfilter von WebSphere eXtreme Scale Version 7.0.

Tabelle 2. Laufzeitdateien für WebSphere eXtreme Scale Client. WebSphere eXtreme Scale Client stützt sich auf ObjectGrid-Prozesse und zugehörige APIs. In der folgenden Tabelle sind die JAR-Dateien aufgelistet, die in der Installation enthalten sind.

Dateiname	Umgebung	Installations-position	Beschreibung
wxsdynacache.jar	Client und Server	dynacache/lib	Die Datei wxsdynacache.jar enthält die erforderlichen Klassen für den dynamischen Cacheprovider. Die Datei wird automatisch in die Serverlaufzeitumgebung eingeschlossen, wenn Sie die bereitgestellten Scripts verwenden.
wxshyperic.jar	Dienstprogramm	hyperic/lib	Das Erkennungs-Plug-in von WebSphere eXtreme Scale für den SpringSource-Hyperic-Überwachungsagenten
ogagent.jar	Lokal, Client und Server	lib	Die Datei ogagent.jar enthält die Laufzeitklassen, die für die Ausführung des Java-Instrumentierungsagenten erforderlich sind, der mit der API "EntityManager" verwendet wird.
ogclient.jar	Lokal und Client	lib	Die Datei ogclient.jar enthält nur die lokale Laufzeitumgebung und die Clientlaufzeitumgebung. Sie können diese Datei mit J2SE Version 1.4.2 und höher verwenden.
ogspring.jar	Lokal, Client und Server	lib	Die Datei ogspring.jar lt enthält Unterstützungsklassen für die Integration des SpringSource-Spring-Frameworks.
wsogclient.jar	Lokal und Client	lib	Die Datei wsogclient.jar wird installiert, wenn Sie eine Umgebung verwenden, die WebSphere Application Server Version 6.0.2 und höher verwendet. Diese Datei enthält nur die lokale Laufzeitumgebung und die Clientlaufzeitumgebung.
wxssizeagent.jar	Lokal, Client und Server	lib	Die Datei wxssizeagent.jar wird verwendet, um genauere Größeninformationen zu den Cacheeinträgen bereitzustellen, wenn Java Runtime Environment (JRE) Version 1.5 oder höher verwendet wird.
ibmcfw.jar ibmext.jar ibmorb.jar ibmorbapi.jar	Client und Server	lib/endorsed	Diese Gruppe von Dateien enthält die ORB-Laufzeitumgebung, die für die Ausführung von Anwendungen in Java-SE-Prozessen verwendet wird.
restservice.ear	Client	restservice/lib	Die Datei restservice.ear enthält das Unternehmensarchiv der REST-Datenserviceanwendung von eXtreme Scale für Umgebungen von WebSphere Application Server.
restservice.war	Client	restservice/lib	Die Datei restservice.war enthält das Webarchiv des REST-Datenservice von eXtreme Scale für Anwendungsserver eines anderen Anbieters.
xsadmin.jar	Dienstprogramm	samples	Die Datei xsadmin.jar enthält das Verwaltungsdienstprogramm für die Muster von eXtreme Scale.
sessionobjectgrid.jar	Client und Server	session/lib	Die Datei sessionobjectgrid.jar enthält die Laufzeitumgebung von eXtreme Scale für das HTTP-Sitzungsmanagement.

Tabelle 2. Laufzeitdateien für WebSphere eXtreme Scale Client (Forts.). WebSphere eXtreme Scale Client stützt sich auf ObjectGrid-Prozesse und zugehörige APIs. In der folgenden Tabelle sind die JAR-Dateien aufgelistet, die in der Installation enthalten sind.

Dateiname	Umgebung	Installations-position	Beschreibung
`splicerlistener.jar`	Dienstprogramm	`session/lib`	Die Datei `splicerlistener.jar` enthält das Dienstprogramm "splicer" für den HTTP-Sitzungs-Listener von eXtreme Scale Version 7.1.
`splicer.jar`	Dienstprogramm	`legacy/ session/lib`	Das Dienstprogramm "splicer" für den HTTP-Sitzungsmanagerfilter von WebSphere eXtreme Scale Version 7.0.

Vorgehensweise

1. Verwenden Sie den Assistenten, um die Installation durchzuführen.
   - Führen Sie das folgende Script aus, um den Assistenten für die vollständige Installation von WebSphere eXtreme Scale zu starten:
     - Linux UNIX *DVD-Stammverzeichnis*`/install`
     - Windows *DVD-Stammverzeichnis*`\install.bat`
   - Führen Sie das folgende Script aus, um den Assistenten für die Installation von WebSphere eXtreme Scale Client zu starten.
     - Linux UNIX `root/`*WXS-Client*`/install`
     - Windows `root\`*WXS-Client*`\install.bat`
2. Folgen Sie der Bedienerführung des Assistenten, und klicken Sie anschließend auf **Fertig stellen**.

   Einschränkung: In der Anzeige mit den optionalen Features werden die Features aufgelistet, die Sie installieren können. Features können der Produktumgebung nach der Produktinstallation jedoch nicht einzeln hinzugefügt werden. Wenn Sie sich während der Erstinstallation des Produkts gegen die Installation eines Features entscheiden, müssen Sie das Produkt deinstallieren und erneut installieren, um das Feature hinzuzufügen.

Ergebnisse

Windows Wenn Sie WebSphere eXtreme Scale Client unter Windows® installieren, finden Sie in den Ergebnissen der Installation möglicherweise den folgenden Text:

```
Erfolg: Die Installation des folgenden Produkts war erfolgreich:
WebSphere eXtreme Scale Client. In einigen Konfigurationsschritten sind Fehler
aufgetreten. Weitere Informationen finden Sie in der folgenden Protokolldatei:
<WAS-Installationsstammverzeichnis>\logs\wxs_client\install\log.txt"
Sehen Sie sich das Installationsprotokoll (log.txt) und das Erweiterungsprotokoll
des Deployment Manager an.
```

Wenn Sie einen Fehler sehen, der sich auf die Datei `iscdeploy.sh` bezieht, können Sie diesen ignorieren. Dieser Fehler verursacht keine Probleme.

Nächste Schritte

Lesen Sie die Informationen zur Konfiguration von eXtreme Scale, um Ihre Clientanwendungsprozesse und Serverprozesse zu konfigurieren.

WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client mit WebSphere Application Server integrieren

Sie können WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client in einer Umgebung installieren, in der WebSphere Application Server oder WebSphere Application Server Network Deployment installiert ist. Sie können die vorhandenen Features von WebSphere Application Server oder WebSphere Application Server Network Deployment verwenden, um die Anwendungen von eXtreme Scale zu erweitern.

Vorbereitende Schritte

- Installieren Sie WebSphere Application Server oder WebSphere Application Server Network Deployment. Weitere Informationen finden Sie unter "Anwendungsserverumgebung installieren".
- Wenden Sie je nach Version, die Sie installieren (Version 6.0.x, Version 6.1 oder Version 7.0), das neueste Fixpack für WebSphere Application Server bzw. WebSphere Application Server Network Deployment an, um den Änderungsstand zu aktualisieren. Weitere Informationen finden Sie im Information-Center-Artikel "Aktuelle Fixpacks für WebSphere Application Server".
- Vergewissern Sie sich, dass das Zielinstallationsverzeichnis keine vorhandene Installation von WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client enthält.
- Stoppen Sie alle Prozesse, die in Ihrer Umgebung mit WebSphere Application Server oder WebSphere Application Server Network Deployment aktiv sind. Weitere Informationen zu den Befehlen "stopManager", "stopNode" und "stopServer" finden Sie unter "wsadmin-Scripting-Befehlszeilentools verwenden".

  Vorsicht:
  Stellen Sie sicher, dass alle aktiven Prozesse gestoppt sind. Wenn die aktiven Prozesse nicht gestoppt sind, wird die Installation fortgesetzt, was zu unvorhersehbaren Ereignissen führen und die Installation auf einigen Plattformen in einem unbestimmten Zustand hinterlassen kann.

Wichtig: Wenn Sie WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client installieren, müssen Sie dasselbe Verzeichnis verwenden, in dem Sie auch WebSphere Application Server installiert haben. Wenn Sie WebSphere Application Server beispielsweise in `C:\<WAS-Stammverzeichnis>` installiert haben, müssen Sie `C:<WAS-Stammverzeichnis>` auch als Zielverzeichnis für Ihre Installation von WebSphere eXtreme Scale bzw. WebSphere eXtreme Scale Client auswählen.

Informationen zu diesem Vorgang

Integrieren Sie eXtreme Scale mit WebSphere Application Server oder WebSphere Application Server Network Deployment, um die Features von eXtreme Scale auf Ihre Java-EE-Anwendungen (Java Platform, Enterprise Edition) anzuwenden. Java-EE-Anwendungen enthalten Daten-Grids und greifen auf die Daten-Grids über eine Clientverbindung zu.

Tabelle 3. Laufzeitdateien für WebSphere eXtreme Scale. In der folgenden Tabelle sind die JAR-Dateien aufgelistet, die in der Installation enthalten sind.

Dateiname	Umgebung	Installations-position	Beschreibung
`wxsdynacache.jar`	Client und Server	`lib`	Die Datei `wxsdynacache.jar` enthält die erforderlichen Klassen für den dynamischen Cacheprovider.

Tabelle 3. Laufzeitdateien für WebSphere eXtreme Scale (Forts.). In der folgenden Tabelle sind die JAR-Dateien aufgelistet, die in der Installation enthalten sind.

Dateiname	Umgebung	Installations-position	Beschreibung
wsobjectgrid.jar	Lokal und Client	lib	Die Datei wsobjectgrid.jar enthält die lokale, die Client- und die Serverlaufzeitumgebung von eXtreme Scale.
ogagent.jar	Lokal, Client und Server	lib	Die Datei ogagent.jar enthält die Laufzeitklassen, die für die Ausführung des Java-Instrumentierungsagenten erforderlich sind, der mit der API "EntityManager" verwendet wird.
ogsip.jar	Server	lib	Die Datei ogsip.jar enthält die eXtreme-Scale-Laufzeitumgebung für das SIP-Sitzungsmanagement (Session Initiation Protocol), die mit WebSphere Application Server Version 6.1.x kompatibel ist.
sessionobjectgrid.jar	Client und Server	lib	Die Datei sessionobjectgrid.jar enthält die Laufzeitumgebung von eXtreme Scale für das HTTP-Sitzungsmanagement.
sessionobjectgridsip.jar	Server	lib	Die Datei sessionobjectgridsip.jar enthält die Laufzeitumgebung von eXtreme Scale für das SIP-Sitzungsmanagement, die mit WebSphere Application Server Version 7.x kompatibel ist.
wsogclient.jar	Lokal und Client	lib	Die Datei wsogclient.jar wird installiert, wenn Sie eine Umgebung verwenden, die WebSphere Application Server Version 6.0.2 und höher verwendet. Diese Datei enthält nur die lokale Laufzeitumgebung und die Clientlaufzeitumgebung.
wxssizeagent.jar	Lokal, Client und Server	lib	Die Datei wxssizeagent.jar wird verwendet, um genauere Größeninformationen zu den Cacheeinträgen bereitzustellen, wenn Java Runtime Environment (JRE) Version 1.5 oder höher verwendet wird.
oghibernate-cache.jar	Client und Server	optionalLibraries/ObjectGrid	Die Datei oghibernate-cache.jar enthält das L2-Cache-Plug-in von eXtreme Scale für JBoss Hibernate.
ogspring.jar	Lokal, Client und Server	optionalLibraries/ObjectGrid	Die Datei ogspring.jar lt enthält Unterstützungsklassen für die Integration des SpringSource-Spring-Frameworks.
xsadmin.jar	Dienstprogramm	optionalLibraries/ObjectGrid	Die Datei xsadmin.jar enthält das Verwaltungsdienstprogramm für die Muster von eXtreme Scale.
ibmcfw.jar ibmext.jar ibmorb.jar ibmorbapi.jar	Client und Server	optionalLibraries/ObjectGrid/endorsed	Diese Gruppe von Dateien enthält die ORB-Laufzeitumgebung, die für die Ausführung von Anwendungen in Java-SE-Prozessen verwendet wird.
wxshyperic.jar	Dienstprogramm	optionalLibraries/ObjectGrid/hyperic/lib	Das Erkennungs-Plug-in von WebSphere eXtreme Scale für den SpringSource-Hyperic-Überwachungsagenten
restservice.ear	Client	optionalLibraries/ObjectGrid/restservice/lib	Die Datei restservice.ear enthält das Unternehmensarchiv der REST-Datenserviceanwendung von eXtreme Scale für Umgebungen von WebSphere Application Server.
restservice.war	Client	optionalLibraries/ObjectGrid/restservice/lib	Die Datei restservice.war enthält das Webarchiv des REST-Datenservice von eXtreme Scale für Anwendungsserver eines anderen Anbieters.
splicerlistener.jar	Dienstprogramm	optionalLibraries/ObjectGrid/session/lib	Die Datei splicerlistener.jar enthält das Splicer-Dienstprogramm für den Filter von eXtreme Scale für den HTTP-Sitzungsmanager.
splicer.jar	Dienstprogramm	optionalLibraries/ObjectGrid/legacy/session/lib	Die Datei splicer.jar enthält das Splicer-Dienstprogramm der Version 7.0 für den HTTP-Sitzungsmanagerfilter von eXtreme Scale.

Tabelle 4. Laufzeitdateien für WebSphere eXtreme Scale Client. In der folgenden Tabelle sind die JAR-Dateien aufgelistet, die in der Installation enthalten sind.

Dateiname	Umgebung	Installations-position	Beschreibung
wxsdynacache.jar	Client und Server	lib	Die Datei wxsdynacache.jar enthält die erforderlichen Klassen für den dynamischen Cacheprovider.
ogagent.jar	Lokal, Client und Server	lib	Die Datei ogagent.jar enthält die Laufzeitklassen, die für die Ausführung des Java-Instrumentierungsagenten erforderlich sind, der mit der API "EntityManager" verwendet wird.
ogsip.jar	Server	lib	Die Datei ogsip.jar enthält die eXtreme-Scale-Laufzeitumgebung für das SIP-Sitzungsmanagement (Session Initiation Protocol), die mit WebSphere Application Server Version 6.1.x kompatibel ist.
sessionobjectgrid.jar	Client und Server	lib	Die Datei sessionobjectgrid.jar enthält die Laufzeitumgebung von eXtreme Scale für das HTTP-Sitzungsmanagement.
sessionobjectgridsip.jar	Server	lib	Die Datei sessionobjectgridsip.jar enthält die Laufzeitumgebung von eXtreme Scale für das SIP-Sitzungsmanagement, die mit WebSphere Application Server Version 7.x kompatibel ist.
wsogclient.jar	Lokal und Client	lib	Die Datei wsogclient.jar wird installiert, wenn Sie eine Umgebung verwenden, die WebSphere Application Server Version 6.0.2 und höher verwendet. Diese Datei enthält nur die lokale Laufzeitumgebung und die Clientlaufzeitumgebung.
wxssizeagent.jar	Lokal, Client und Server	lib	Die Datei wxssizeagent.jar wird verwendet, um genauere Größeninformationen zu den Cacheeinträgen bereitzustellen, wenn Java Runtime Environment (JRE) Version 1.5 oder höher verwendet wird.
oghibernate-cache.jar	Client und Server	optionalLibraries/ObjectGrid	Die Datei oghibernate-cache.jar enthält das L2-Cache-Plug-in von eXtreme Scale für JBoss Hibernate.
ogspring.jar	Lokal, Client und Server	optionalLibraries/ObjectGrid	Die Datei ogspring.jar lt enthält Unterstützungsklassen für die Integration des SpringSource-Spring-Frameworks.
xsadmin.jar	Dienstprogramm	optionalLibraries/ObjectGrid	Die Datei xsadmin.jar enthält das Verwaltungsdienstprogramm für die Muster von eXtreme Scale.
ibmcfw.jar ibmext.jar ibmorb.jar ibmorbapi.jar	Client und Server	optionalLibraries/ObjectGrid/endorsed	Diese Gruppe von Dateien enthält die ORB-Laufzeitumgebung, die für die Ausführung von Anwendungen in Java-SE-Prozessen verwendet wird.
wxshyperic.jar	Dienstprogramm	optionalLibraries/ObjectGrid/hyperic/lib	Das Erkennungs-Plug-in von WebSphere eXtreme Scale für den SpringSource-Hyperic-Überwachungsagenten
restservice.ear	Client	optionalLibraries/ObjectGrid/restservice/lib	Die Datei restservice.ear enthält das Unternehmensarchiv der REST-Datenserviceanwendung von eXtreme Scale für Umgebungen von WebSphere Application Server.
restservice.war	Client	optionalLibraries/ObjectGrid/restservice/lib	Die Datei restservice.war enthält das Webarchiv des REST-Datenservice von eXtreme Scale für Anwendungsserver eines anderen Anbieters.
splicerlistener.jar	Dienstprogramm	optionalLibraries/ObjectGrid/session/lib	Die Datei splicerlistener.jar enthält das Splicer-Dienstprogramm für den Filter von eXtreme Scale für den HTTP-Sitzungsmanager.
splicer.jar	Dienstprogramm	optionalLibraries/ObjectGrid/legacy/session/lib	Die Datei splicer.jar enthält das Splicer-Dienstprogramm der Version 7.0 für den HTTP-Sitzungsmanagerfilter von eXtreme Scale.

Vorgehensweise

1. Verwenden Sie den Assistenten, um die Installation durchzuführen.

- Führen Sie das folgende Script aus, um den Assistenten für die vollständige Installation von WebSphere eXtreme Scale zu starten:
  - Linux UNIX *DVD-Stammverzeichnis*/install
  - Windows *DVD-Stammverzeichnis*\install.bat
- Führen Sie das folgende Script aus, um den Assistenten für die Installation von WebSphere eXtreme Scale Client zu starten.
  - Linux UNIX root/*WXS-Client*/install
  - Windows root*WXS-Client*\install.bat

2. Folgen Sie den Eingabeaufforderungen im Assistenten.

   In der Anzeige mit den optionalen Features werden die Features aufgelistet, die Sie installieren können. Features können der Produktumgebung nach der Produktinstallation jedoch nicht einzeln hinzugefügt werden. Wenn Sie sich während der Erstinstallation des Produkts gegen die Installation eines Features entscheiden, müssen Sie das Produkt deinstallieren und erneut installieren, um das Feature hinzuzufügen.

   In der Anzeige "Profilerweiterung" werden die vorhandenen Profile aufgelistet, die Sie mit den Features von eXtreme Scale erweitern können. Wenn Sie vorhandene Profile auswählen, die bereits im Gebrauch sind, erscheint eine Warnanzeige. Zum Fortsetzen der Installation müssen Sie die in den Profilen konfigurierten Server stoppen oder auf **Zurück** klicken, um die Profile aus Ihrer Auswahl zu entfernen.

Ergebnisse

Windows Wenn Sie WebSphere eXtreme Scale Client unter Windows installieren, finden Sie in den Ergebnissen der Installation möglicherweise den folgenden Text:

```
Erfolg: Die Installation des folgenden Produkts war erfolgreich:
WebSphere eXtreme Scale Client. In einigen Konfigurationsschritten sind Fehler
aufgetreten. Weitere Informationen finden Sie in der folgenden Protokolldatei:
<WAS-Installationsstammverzeichnis>\logs\wxs_client\install\log.txt"
Sehen Sie sich das Installationsprotokoll (log.txt) und das Erweiterungsprotokoll
des Deployment Manager an.
```

Wenn Sie einen Fehler sehen, der sich auf die Datei `iscdeploy.sh` bezieht, können Sie diesen ignorieren. Dieser Fehler verursacht keine Probleme.

Nächste Schritte

Wenn Sie WebSphere Application Server Version 6.1 oder Version 7.0 ausführen, können Sie das PMT-Plug-in (Profile Management Tool) oder den Befehl "manageprofiles" verwenden. Wenn Sie WebSphere Application Server Version 6.0.2 ausführen, müssen Sie zum Erstellen und Erweitern von Profilen den Befehl "wasprofile" verwenden.

Implementieren Sie Ihre Anwendung, starten Sie einen Katalogservice, und starten Sie die Container in Ihrer Umgebung mit WebSphere Application Server. Weitere Informationen finden Sie im Abschnitt „WebSphere eXtreme Scale mit WebSphere Application Server verwalten” auf Seite 348.

Installation-Factory-Plug-in zum Erstellen und Installieren angepasster Pakete verwenden

Verwenden Sie das IBM Installation-Factory-Plug-in für WebSphere eXtreme Scale, um ein angepasstes Installationspaket (CIP, Customized Installation Package) oder

ein integriertes Installationspaket (IIP, Integrated Installation Package) zu erstellen. Ein angepasstes Installationspaket enthält ein Installationspaket für ein Einzelprodukt und verschiedene optionale Assets. Ein integriertes Installationspaket kombiniert ein oder mehrere Installationspakete zu einem einzigen Installations-Workflow, den Sie entwerfen können.

Vorbereitende Schritte

Bevor Sie angepasste Pakete für eXtreme Scale erstellen, müssen Sie die folgenden Produkte herunterladen:

- IBM Installation Factory for WebSphere Application Server
- IBM Installation-Factory-Plug-in für WebSphere eXtreme Scale

Informationen zu diesem Vorgang

Mit Installation Factory können Sie ein angepasstes Installationspaket erstellen, indem Sie eine einzelne Produktkomponente mit Wartungspaketen, Anpassungsscripts und anderen Dateien kombinieren. Wenn Sie ein integriertes Installationspaket erstellen, fassen Sie einzelne Komponenten oder Installationspakete zu einem einzelnen Installationspaket zusammen.

Build-Definitionsdatei

Eine Build-Definitionsdatei ist ein XML-Dokument, in dem spezifiziert wird, wie ein angepasstes Installationspaket (CIP, Customized Installation Package) oder integriertes Installationspaket (IIP, Integrated Installation Package) erstellt und installiert wird. IBM Installation Factory for WebSphere eXtreme Scale liest die Paketdetails in der Build-Definitionsdatei, um ein angepasstes Installationspaket bzw. integriertes Installationspaket zu generieren.

Damit Sie ein angepasstes Installationspaket oder ein integriertes Installationspaket erstellen können, müssen Sie für jedes angepasste Paket eine Build-Definitionsdatei erstellen. In der Build-Definitionsdatei wird beschrieben, welche Produktkomponenten oder Installationspakete installiert werden sollen. Außerdem enthält diese Datei Angaben zur Position des angepassten Installationspakets bzw. integrierten Installationspakets, zu den einzuschließenden Wartungspaketen, zu den Installationsscripts und anderen ausgewählten Dateien, die in das Paket eingeschlossen werden sollen. Außerdem können Sie in der Build-Definitionsdatei für das integrierte Installationspaket die Reihenfolge festlegen, in der Installation Factory die Installationspakete installiert.

Der Assistent für Build-Definition führt Sie schrittweise durch die Erstellung einer Build-Definitionsdatei. Sie können mit dem Assistenten für Build-Definition auch eine vorhandene Build-Definitionsdatei ändern. In jeder Anzeige des Assistenten für Build-Definition werden Sie zur Eingabe von Informationen zu einem angepassten Paket aufgefordert, z. B. der Paketkennung, der Installationsposition für die Build-Definition und der Installationsposition für das angepasste Paket. Alle diese Informationen werden in der neuen Build-Definitionsdatei gespeichert bzw. in einer vorhandenen Build-Definitionsdatei geändert und gespeichert. Weitere Informationen finden Sie in den Abschnitten "Anzeigen des Assistenten für Build-Definition für angepasste Installationspakete" und "Anzeigen des Assistenten für Build-Definition für integrierte Installationspakete".

Wenn Sie nur die Build-Definitionsdatei erstellen möchten, können Sie das Befehlszeilenschnittstellentool ausführen, um das angepasste Paket außerhalb der grafi-

schen Benutzerschnittstelle zu generieren. Weitere Informationen finden Sie im Abschnitt „Angepasstes oder integriertes Installationspaket im unbeaufsichtigten Modus installieren” auf Seite 35.

Build-Definitionsdatei erstellen und ein angepasstes Installationspaket erstellen

Das IBM Installation-Factory-Plug-in für WebSphere eXtreme Scale generiert ein angepasstes Installationspaket entsprechend den Details, die Sie in der Build-Definitionsdatei angeben. Die Build-Definition gibt das zu installierende Produktpaket, die Position des angepassten Installationspakets, die in die Installation einzuschließenden Wartungspakete, die Installationsscriptdateien und alle zusätzlichen Dateien an, die in das angepasste Installationspaket eingeschlossen werden sollen.

Informationen zu diesem Vorgang

Sie können den Assistenten für Build-Definition verwenden, um eine Build-Definitionsdatei zu erstellen und ein angepasstes Installationspaket zu generieren.

Vorgehensweise

1. Führen Sie das folgende Script im Verzeichnis *IF-Ausgangsverzeichnis*/bin aus, um Installation Factory zu starten:
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`

   Klicken Sie auf das Symbol für **Neue Build-Definition**.
2. Wählen Sie das Produkt aus, das in die Build-Definitionsdatei eingeschlossen werden soll, und klicken Sie anschließend auf **Fertig stellen**, um den Assistenten für Build-Definition zu starten.
3. Folgen Sie den Eingabeaufforderungen im Assistenten.

   Klicken Sie in der Anzeige "Installations- und Deinstallationsscripts" auf **Scripts hinzufügen...**, um alle angepassten Installationsscripts in die Tabelle einzutragen. Geben Sie die Position der Scriptdateien ein, und wählen Sie das Kontrollkästchen ab, um fortzufahren, wenn eine Fehlernachricht angezeigt wird. Die Operation wird standardmäßig gestoppt. Klicken Sie auf **OK**, um zur Anzeige zurückzukehren.

Ergebnisse

Sie haben die Build-Definitionsdatei erstellt und angepasst und das angepasste Installationspaket generiert, wenn Sie sich für die Arbeit im Modus "Verbunden" entschieden haben.

Wenn der Assistent für Build-Definition Ihnen die Option zum Generieren des angepassten Installationspakets aus der Build-Definitionsdatei nicht anbietet, können Sie das Paket trotzdem generieren, indem Sie das Script `ifcli.sh|bat` im Verzeichnis *IF-Ausgangsverzeichnis*/bin ausführen.

Nächste Schritte

Installieren Sie das angepasste Installationspaket. Weitere Informationen finden Sie unter „Angepasstes Installationspaket installieren”.

Angepasstes Installationspaket installieren:

Sie können den Produktinstallationsprozess vereinfachen, indem Sie ein angepasstes Installationspaket (CIP, Customized Installation Package) installieren. Ein angepasstes Installationspaket ist ein Installations-Image für ein einziges Produkt, das ein oder mehrere Pakete, Konfigurationsscripts und andere Dateien enthalten kann.

Vorbereitende Schritte

Damit Sie ein integriertes Installationspaket installieren können, müssen Sie zuerst eine Build-Definitionsdatei erstellen, in der Sie die in das integrierte Installationspaket einzuschließenden Optionen angeben. Weitere Informationen finden Sie im Abschnitt „Build-Definitionsdatei erstellen und ein angepasstes Installationspaket erstellen" auf Seite 28.

Informationen zu diesem Vorgang

Mit einem angepassten Installationspaket kann eine einzelne Produktkomponente mit Wartungspaketen, Anpassungsscripts und anderen Dateien kombiniert und installiert werden.

Vorgehensweise

1. Stoppen Sie alle Prozesse, die auf der Workstation ausgeführt werden, die Sie für die Installation vorbereiten. Zum Stoppen des Deployment Manager führen Sie das folgende Script aus:
   - Linux UNIX *Profilstammverzeichnis*/bin/stopManager.sh
   - Windows *Profilstammverzeichnis*\bin\stopManager.bat

   Zum Stoppen der Knoten führen Sie das folgende Script aus:
   - Linux UNIX *Profilstammverzeichnis*/bin/stopNode.sh
   - Windows *Profilstammverzeichnis*\bin\stopNode.bat
2. Führen Sie das folgende Script aus, um die Installation zu starten:
   - Linux UNIX *CIP-Ausgangsverzeichnis*/bin/install
   - Windows *CIP-Ausgangsverzeichnis*\bin\install.bat
3. Folgen Sie den Aufforderungen des Assistenten, um die Installation durchzuführen:

   In der Anzeige mit den optionalen Features werden die Features aufgelistet, die Sie installieren können. Features können der Produktumgebung nach der Produktinstallation jedoch nicht einzeln hinzugefügt werden. Wenn Sie sich während der Erstinstallation des Produkts gegen die Installation eines Features entscheiden, müssen Sie das Produkt deinstallieren und erneut installieren, um das Feature hinzuzufügen.

   In der Anzeige "Profilerweiterung" werden die vorhandenen Profile aufgelistet, die Sie mit den Features von eXtreme Scale erweitern können. Wenn Sie vorhandene Profile auswählen, die bereits im Gebrauch sind, erscheint eine Warnanzeige. Zum Fortsetzen der Installation müssen Sie die in den Profilen konfigurierten Server stoppen oder auf **Zurück** klicken, um die Profile aus Ihrer Auswahl zu entfernen.

Ergebnisse

Sie haben das angepasste Installationspaket ordnungsgemäß installiert.

Nächste Schritte

Wenn Sie WebSphere Application Server Version 6.1 oder Version 7.0 ausführen, können Sie das PMT-Plug-in (Profile Management Tool) oder den Befehl "manageprofiles" verwenden, um Profile zu erstellen und zu erweitern. Wenn Sie WebSphere Application Server Version 6.0.2 ausführen, müssen Sie zum Erstellen und Erweitern von Profilen den Befehl "wasprofile" verwenden. Weitere Informationen finden Sie im Abschnitt „Profile für WebSphere eXtreme Scale erstellen und erweitern" auf Seite 44.

Wenn Sie Profile für eXtreme Scale während des Installationsprozesses erweitert haben, können Sie Anwendungen implementieren, einen Katalogservice starten und die Container in Ihrer Umgebung von WebSphere Application Server starten. Weitere Informationen hierzu finden Sie im Abschnitt „WebSphere eXtreme Scale mit WebSphere Application Server verwalten" auf Seite 348.

Ein angepasstes Installationspaket zum Anwenden von Wartungspaketen auf eine vorhandene Produktinstallation installieren:

Sie können Wartungspakete für eine vorhandene Produktinstallation anwenden, indem Sie ein angepasstes Installationspaket (CIP, Customized Installation Package) installieren. Die Anwendung von Wartungspaketen auf eine vorhandene Installation mit einem angepassten Installationspaket wird häufig als *Slip-Installation* bezeichnet.

Vorbereitende Schritte

Erstellen Sie eine Build-Definitionsdatei, um die in das angepasste Installationspaket einzuschließenden Optionen anzugeben. Weitere Informationen finden Sie im Abschnitt „Build-Definitionsdatei erstellen und ein angepasstes Installationspaket erstellen" auf Seite 28.

Informationen zu diesem Vorgang

Wenn Sie Wartungspakete mit einem angepassten Installationspaket anwenden, das ein Aktualisierungspaket und/oder einen Fixpack enthält, werden alle zuvor installierten APARs (Authorized Program Analysis Report) in dieser Installation vom Assistenten deinstalliert. Wenn das angepasste Installationspaket dieselbe Version hat wie das Produkt, werden die zuvor installierten APARs nur dann nicht deinstalliert, wenn sie im angepassten Installationspaket enthalten sind. Für eine erfolgreiche Anwendung von Wartungspaketen auf eine vorhandene Installation müssen Sie die installierten Features im angepassten Installationspaket anwenden.

Vorgehensweise

1. Stoppen Sie alle Prozesse, die auf der Workstation ausgeführt werden, die Sie für die Installation vorbereiten. Zum Stoppen des Deployment Manager führen Sie das folgende Script aus:
   - Linux UNIX *Profilstammverzeichnis*/bin/stopManager.sh
   - Windows *Profilstammverzeichnis*\bin\stopManager.bat

   Zum Stoppen der Knoten führen Sie das folgende Script aus:
   - Linux UNIX *Profilstammverzeichnis*\bin\stopNode.sh
   - Windows *Profilstammverzeichnis*\bin\stopNode.bat
2. Führen Sie das folgende Script aus, um die Installation zu starten:

- Linux UNIX *CIP-Ausgangsverzeichnis*/bin/install
- Windows *CIP-Ausgangsverzeichnis*\bin\install.bat

3. Folgen Sie den Aufforderungen des Assistenten, um die Installation durchzuführen:

   Die Zusammenfassung in der Installationsvoranzeige listet die Produktversion und alle Features und vorläufigen Fixes auf, die angewendet werden sollen. Als Nächstes wendet der Assistent die Wartung erfolgreich an und aktualisiert die Features des Produkts.

Ergebnisse

Die Produktbinärdateien werden in das Verzeichnis *WAS-Ausgangsverzeichnis*/properties/version/nif/backup kopiert. Sie können IBM Update Installer verwenden, um die Aktualisierung zu deinstallieren und die Workstation wiederherzustellen. Weitere Informationen finden Sie im Abschnitt „CIP-Aktualisierungen aus einer vorhandenen Produktinstallation deinstallieren".

CIP-Aktualisierungen aus einer vorhandenen Produktinstallation deinstallieren:

Sie können CIP-Aktualisierungen (Customized Installation Package, angepasstes Installationspaket) aus einer vorhandenen Produktinstallation deinstallieren, ohne das gesamte Produkt deinstallieren zu müssen. Verwenden Sie IBM Update Installer Version 7.0.0.4, um CIP-Aktualisierungen zu deinstallieren. Diese Task wird auch als *Slip-Deinstallation* bezeichnet.

Vorbereitende Schritte

Es muss mindestens eine Kopie des Produkts auf dem System installiert sein.

Vorgehensweise

1. Laden Sie Version 7.0.0.4 von Update Installer von der folgenden FTP-Site herunter:

   ftp://ftp.software.ibm.com/software/websphere/cw/process_server/FEP/UPDI/7004
2. Installieren Sie Update Installer. Weitere Informationen finden Sie im Artikel "Update Installer für WebSphere Software" im Information Center von WebSphere Application Server.
3. Deinstallieren Sie alle Fixpacks, Refresh-Packs und vorläufigen Fixes, die Sie der Umgebung nach der Installation des angepassten Installationspakets hinzugefügt haben.
4. Deinstallieren Sie alle vorläufigen Fixes, die Sie in die Slip-Installation eingeschlossen haben. Diese Vorgehensweise ist dieselbe wie bei der Deinstallation eines einzelnen Fixpacks oder Refresh-Packs. Die Wartung, die im angepassten Installationspaket enthalten war, ist jetzt allerdings in einer einzelnen Operation enthalten.
5. Deinstallieren Sie das angepasste Installationspaket mit Update Installer. Die Wartungsstufen werden auf den Stand vor Aktualisierung zurückgesetzt, und das angepasste Installationspaket wird mit der CIP-ID gekennzeichnet, die dem Dateinamen als Präfix hinzugefügt wird. Das folgende Beispiel veranschaulicht, das ein angepasstes Installationspaket anders als andere reguläre Wartungspakete in der Anzeige zur Auswahl der Wartungspakete angezeigt wird:

   Angepasstes Installationspaket

   com.ibm.ws.cip.7000.wxs.primary.ext.pak

Ergebnisse

Die CIP-Aktualisierungen wurden ordnungsgemäß aus einer vorhandenen Produktinstallation entfernt.

Build-Definitionsdatei erstellen und integriertes Installationspaket generieren

Das IBM Installation-Factory-Plug-in für WebSphere eXtreme Scale generiert ein integriertes Installationspaket, das auf den Eigenschaften basiert, die in der Build-Definitionsdatei bereitgestellt werden. Die Build-Definitionsdatei enthält Informationen, wie z. B. Informationen zu den in das integrierte Installationspaket einzuschließenden Paketen, zur Reihenfolge, in der Installation Factory die Pakete installiert, und zur Position des integrierten Installationspakets.

Informationen zu diesem Vorgang

Sie können den Assistenten für Build-Definition verwenden, um eine Build-Definitionsdatei zu erstellen und ein integriertes Installationspaket zu generieren.

Vorgehensweise

1. Führen Sie das folgende Script im Verzeichnis *IF-Ausgangsverzeichnis*/bin aus, um Installation Factory zu starten:
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`
2. Klicken Sie auf das Symbol **Neues integriertes Installationspaket erstellen**, um den Assistenten für Build-Definition zu starten.
3. Folgen Sie den Eingabeaufforderungen im Assistenten.
   a. Wählen Sie in der Anzeige "Integriertes Installationspaket erstellen" in der Liste ein unterstütztes Installationspaket aus, und klicken Sie auf **Installationsprogramm hinzufügen**, um das Installationspaket dem integrierten Installationspaket hinzuzufügen. Es erscheint eine Anzeige, in der der Paketname, die Paketkennung und die Paketeigenschaften angezeigt werden. Wenn Sie bestimmte Informationen zum ausgewählten Paket anzeigen möchten, klicken Sie auf **Informationen zum Installationspaket anzeigen**. Klicken Sie auf **Ändern**, um den Verzeichnispfad zum Installationspaket für jedes Betriebssystem einzugeben. Wenn Sie ein Installationspaket für WebSphere Extended Deployment hinzufügen, wählen Sie das Kontrollkästchen aus, das Ihnen die Möglichkeit gibt, dasselbe Paket für alle unterstützten Betriebssysteme zu verwenden. Klicken Sie auf **OK**, und kehren Sie in die Anzeige "Integriertes Installationspaket erstellen" zurück. Es wird standardmäßig ein Aufruf erstellt.
      - Wenn Sie den Verzeichnispfad zu einem Installationspaket ändern möchten, wählen Sie das Paket in der Liste "In diesem integrierten Installationspaket verwendete Installationspakete" aus, und klicken Sie auf **Ändern**.
      - Wenn Sie einen Aufruf ändern möchten, wählen Sie den Aufruf aus, und klicken Sie auf **Ändern**. Geben Sie die Standardinstallationsposition für den Aufruf auf jedem Betriebssystem an. Geben Sie die Position zur Antwortdatei an, wenn Sie eine unbeaufsichtigte Installation als Standardinstallationsmodus auswählen.
      - Klicken Sie auf **Aufruf hinzufügen**, um dem Installationspaket einen Aufrufbeitrag hinzuzufügen. Es erscheint eine Anzeige, in der Sie die Eigenschaften für den Aufruf eingeben können.

- Klicken Sie auf **Entfernen**, um Installationspakete oder Aufrufe zu entfernen.

4. Sehen Sie sich die Zusammenfassung der von Ihnen ausgewählten Optionen an, wählen Sie die Option **Build-Definitionsdatei speichern und integriertes Installationspaket generieren** aus, und klicken Sie anschließend auf **Fertig stellen**.

   Alternativ können Sie die Build-Definitionsdatei auch ohne Generierung des integrierten Installationspakets speichern. Mit dieser Option generieren Sie das integrierte Installationspaket außerhalb des Assistenten, indem Sie das Script `ifcli.bat | ifcli.sh` im Verzeichnis "*IF-Ausgangsverzeichnis*/bin/" ausführen.

Ergebnisse

Sie haben die Build-Definitionsdatei für ein integriertes Installationspaket erstellt und angepasst.

Nächste Schritte

Installieren Sie das integrierte Installationspaket.

Integriertes Installationspaket installieren:

Verwenden Sie das IBM Installation-Factory-Plug-in für WebSphere eXtreme Scale, um ein integriertes Installationspaket (IIP, Integrated Installation Package) zu installieren. Ein integriertes Installationspaket kombiniert ein oder mehrere Installationspakete zu einem einzigen Workflow, den Sie entwerfen können.

Vorbereitende Schritte

Damit Sie ein integriertes Installationspaket installieren können, müssen Sie zuerst eine Build-Definitionsdatei erstellen, in der Sie die in das integrierte Installationspaket einzuschließenden Optionen angeben. Weitere Informationen finden Sie im Abschnitt „Build-Definitionsdatei erstellen und integriertes Installationspaket generieren” auf Seite 32.

Informationen zu diesem Vorgang

Ein integriertes Installationspaket kann ein oder mehrere allgemein verfügbare Installationspakete, ein oder mehrere angepasste Installationspakete (CIP, Customized Installation Packages) und weitere optionale Dateien und Verzeichnisse enthalten. Indem Sie ein integriertes Installationspaket installieren, fassen Sie mehrere Installationspakete oder *Beiträge* zu einem einem einzelnen Paket zusammen und installieren die Beiträge in einer bestimmte Reihenfolge in einer End-to-End-Installation.

Vorgehensweise

1. Führen Sie das folgende Script aus, um den Assistenten zu starten:
   - Linux UNIX `IIP-Ausgangsverzeichnis/bin/install`
   - Windows `IIP-Ausgangsverzeichnis\bin\install.bat`
2. Klicken Sie in der Eingangsanzeige auf **Produktinfo**, um die Details zum integrierten Installationspaket anzuzeigen, wie z. B. die Paketkennung, die unterstützten Betriebssysteme und die eingeschlossenen Installationspakete.

   Optional: Wenn Sie die Installationsoptionen der einzelnen Pakete ändern möchten, klicken Sie auf **Ändern**.

Optional: Auf der Assistentenseite werden zwei Schaltflächen **Protokoll anzeigen** angezeigt. Wenn Sie die Protokolle der einzelnen Pakete anzeigen möchten, klicken Sie auf die Schaltfläche **Protokoll anzeigen**, die neben der Tabelle angezeigt wird, in der die Installationspakete aufgelistet sind. Klicken Sie auf die Schaltfläche **Protokoll anzeigen**, die neben den Statusinformationen angezeigt wird, um die allgemeinen Protokolldetails des integrierten Installationspakets anzuzeigen.

3. Wählen Sie die auszuführenden Installationspakete aus, und klicken Sie auf **Installieren**. Es wird eine nach Aufruf sortierte Liste aller Beiträge angezeigt, die im integrierten Installationspaket enthalten sind. Wenn ein Beitragsaufruf während der Installation nicht ausgeführt werden soll, wählen Sie das Kontrollkästchen neben dem Feld **Installationsname** ab.

Ergebnisse

Sie haben erfolgreich ein integriertes Installationspaket installiert.

Eine vorhandene Build-Definitionsdatei für ein integriertes Installationspaket ändern:

Sie können die Eigenschaften eines integrierten Installationspakets bearbeiten oder weitere Eigenschaften hinzufügen, um die Installation weiter anzupassen.

Informationen zu diesem Vorgang

Wenn Sie die Eigenschaften eines integrierten Installationspakets ändern möchten, ändern Sie die vorhandene Build-Definitionsdatei.

Vorgehensweise

1. Führen Sie das folgende Script im Verzeichnis *IF-Ausgangsverzeichnis*/bin aus, um Installation Factory zu starten:
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`
2. Klicken Sie auf das Symbol **Build-Definition öffnen**, und wählen Sie die zu änderende Build-Definitionsdatei aus.
3. Wählen Sie die Eigenschaften des integrierten Installationspakets aus, die Sie ändern möchten. Die folgende Liste enthält die möglichen Änderungen, die Sie vornehmen können:
   - Aktuelle Modusauswahl ändern. Im Modus "Verbunden" erstellen Sie eine Build-Definitionsdatei für die Verwendung auf Ihrer Workstation und generieren optional ein integriertes Installationspaket. Im Modus "Nicht verbunden" erstellen Sie die Build-Definitionsdatei für die Verwendung auf einer anderen Workstation.
   - Vom integrierten Installationspaket unterstützte Betriebssysteme hinzufügen oder entfernen.
   - Vorhandene Kennung und Version für das integrierte Installationspaket bearbeiten.
   - Zielposition für die Build-Definitionsdatei bearbeiten.
   - Zielposition für das integrierte Installationspaket bearbeiten.
   - Festlegen, ob ein Installationsassistent für das integrierte Installationspaket angezeigt wird. Im Installationsassistenten werden Informationen zum integrierten Installationspaket und die Installationsoptionen für die Ausführung des integrierten Installationspakets angezeigt.

- Im integrierten Installationspaket enthaltene Installationspaket entfernen, bearbeiten oder neue Installationspakete hinzufügen.

  Wichtig: Wenn Sie ein unterstütztes Betriebssystem hinzugefügt und die Eigenschaften des Installationspaket im integrierten Installationspaket nicht aktualisiert haben, empfangen Sie eine Warnung, in der Sie darauf hingewiesen werden, dass die ausgewählten Beiträge keine Installationspakete enthalten, die für alle Betriebssysteme angegeben sind, die vom integrierten Installationspaket unterstützt werden. Klicken Sie auf **Ja**, um fortzufahren, oder klicken Sie auf **Nein**, um das Installationspaket zu bearbeiten.

4. Sehen Sie sich die Zusammenfassung der von Ihnen ausgewählten Optionen an, wählen Sie die Option **Build-Definitionsdatei speichern und integriertes Installationspaket generieren** aus, und klicken Sie anschließend auf **Fertig stellen**.

Angepasstes oder integriertes Installationspaket im unbeaufsichtigten Modus installieren

Sie können ein angepasstes Installationspaket (Customized Installation Package) oder integriertes Installationspaket (IIP, Integrated Installation Package) für ein Produkt im unbeaufsichtigten Modus installieren, indem Sie entweder eine vollständig qualifizierte Antwortdatei, die Sie speziell auf Ihre Anforderungen konfigurieren, oder Parameter verwenden, die Sie an die Befehlszeile übergeben.

Vorbereitende Schritte

Erstellen Sie die Build-Definitionsdatei für das angepasste Installationspaket bzw. das integrierte Installationspaket. Weitere Informationen finden Sie im Abschnitt „Build-Definitionsdatei erstellen und ein angepasstes Installationspaket erstellen" auf Seite 28.

Informationen zu diesem Vorgang

Bei einer unbeaufsichtigten Installation wird dasselbe Installationsprogramm verwendet, das auch bei der Installation über die grafische Benutzerschnittstelle verwendet wird. Anstatt jedoch eine Assistentenschnittstelle anzuzeigen, werden bei der unbeaufsichtigten Installation alle Antworten aus einer Datei, die Sie anpassen, oder aus Parametern gelesen, die Sie an die Befehlszeile übergeben. Wenn Sie ein integriertes Installationspaket im unbeaufsichtigten Modus installieren, können Sie einen Beitrag mit einer Kombination von Optionen, die Sie direkt in der Befehlszeile angeben, und Optionen, die Sie in einer Antwortdatei angeben, aufrufen. Alle Beitragsoptionen, die Sie an die Befehlszeile übergeben, bewirken jedoch, dass das Installationsprogramm des integrierten Installationspakets alle Optionen ignoriert, die in der Antwortdatei eines bestimmten Beitrags angegeben sind. Ausführliche Informationen finden Sie im Abschnitt "Installationsoptionen für integrierte Installationspakete".

Anmerkung: Sie müssen den vollständig qualifizierten Namen der Antwortdatei angeben. Wenn Sie den relativen Pfad angeben, scheitert die Installation ohne Ausgabe einer entsprechenden Fehlernachricht.

Vorgehensweise

1. Optional: Wenn Sie sich für die Installation des angepassten Installationspakets bzw. integrierten Installationspakets über eine Antwortdatei entscheiden, passen Sie zuerst die Datei an.

a. Kopieren Sie die Antwortdatei `wxssetup.response.txt` von der Produkt-DVD auf Ihr Plattenlaufwerk.
b. Öffnen Sie die Antwortdatei mit einem Texteditor Ihrer Wahl, und bearbeiten Sie sie. Die Datei enthält Kommentare, die Sie beim Konfigurationsprozess unterstützen. Sie muss die folgenden Parameter enthalten:
   - die Lizenzvereinbarung,
   - die Position der Produktinstallation.

   Tipp: Das Installationsprogramm verwendet die Position, die Sie für Ihre Installation auswählen, um zu bestimmen, wo Ihre Instanz von WebSphere Application Server installiert ist. Wenn Sie die Installation auf einem Knoten mit mehreren Instanzen von WebSphere Application Server durchführen, müssen Sie die Position klar definieren.
c. Führen Sie das folgende Script aus, um die angepasste Antwortdatei zu starten.
   - Linux UNIX `install -options /absoluter_Pfad/Antwortdatei.txt -silent`
   - Windows `install.bat -options C:\Laufwerkspfad\Antwortdatei.txt -silent`

2. Optional: Wenn Sie sich für die Installation des angepassten bzw. integrierten Installationspakets durch Übergabe bestimmter Parameter in der Befehlszeile entscheiden, führen Sie das folgende Script aus, um die Installation zu starten:
   - Linux UNIX `install -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=Installationsposition`
   - Windows `install.bat -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=Installationsposition`

   Installationsposition steht für die Position der vorhandenen Installation von WebSphere Application Server.
3. Suchen Sie in den während der Installation erstellten Protokollen nach Fehlern oder Installationsfehlern.

Ergebnisse

Sie haben das angepasste Installationspaket bzw. das integrierte Installationspaket im unbeaufsichtigten Modus installiert.

Nächste Schritte

Wenn Sie WebSphere Application Server Version 6.1 oder Version 7.0 ausführen, können Sie das PMT-Plug-in (Profile Management Tool) oder den Befehl "manageprofiles" verwenden, um Profile zu erstellen und zu erweitern. Wenn Sie WebSphere Application Server Version 6.0.2 ausführen, müssen Sie zum Erstellen und Erweitern von Profilen den Befehl "wasprofile" verwenden.

Wenn Sie Profile für eXtreme Scale während des Installationsprozesses erweitert haben, können Sie Anwendungen implementieren, einen Katalogservice starten und die Container in Ihrer Umgebung von WebSphere Application Server starten. Weitere Informationen hierzu finden Sie im Abschnitt „WebSphere eXtreme Scale mit WebSphere Application Server verwalten” auf Seite 348.

Datei `wxssetup.response.txt`:

Sie können eine vollständig qualifizierte Antwortdatei verwenden, um WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client unbeaufsichtigt zu installieren.

Vorsicht:
Fügen Sie keine abschließenden Schrägstriche wie / oder \ am Ende der Installationspositionspfade hinzu. Diese Pfade werden mit dem Attribut "installLocation" angegeben. Wenn Sie einen Schrägstrich am Ende der Installationsposition hinzufügen, kann die Installation fehlschlagen. Der folgende Pfad würde beispielsweise dazu führen, dass die Installation fehlschlägt:

```
-OPT installLocation="/usr/IBM/WebSphere/eXtremeScale/"
```

Der Pfad muss wie folgt angegeben werden:

```
-OPT installLocation="/usr/IBM/WebSphere/eXtremeScale"
```

Antwortdatei für eine vollständige Installation von WebSphere eXtreme Scale

```
################################################################################
#
# InstallShield-Optionsdatei für IBM WebSphere eXtreme Scale V7.1.0
#
# Name des Assistenten: Install
# Quelle des Assistenten: setup.jar
#
# Diese Datei kann verwendet werden, um den Assistenten "Install" mit den im
# Folgenden angegebenen Optionen zu konfigurieren, wenn der Assistent mit der
# Befehlszeilenoption "-options" ausgeführt wird. Die Dokumentation zu jeder
# Einstellung enthält Informationen zum Ändern des Einstellungswerts.
# Schließen Sie alle Werte in Anführungszeichen ein.
#
# Eine gängige Verwendung für eine Optionsdatei ist die Ausführung des
# Assistenten im unbeaufsichtigten Modus. Auf diese Weise kann der Autor
# der Optionsdatei Assistenteneinstellungen festlegen, ohne den Assistenten
# im grafischen Modus oder im Konsolmodus ausführen zu müssen. Wenn Sie
# diese Optionsdatei für die Ausführung im unbeaufsichtigten Modus verwenden
# möchten, verwenden Sie die folgenden Befehlszeilenargumente bei der
# Ausführung des Assistenten:
#
#    -options "D:\installImage\WXS\wxssetup.response" -silent
#
# Sie müssen den vollständig qualifizierten Namen der Antwortdatei angeben.
#
################################################################################

################################################################################
#
# Lizenz akzeptieren
#
# Gültige Werte:
#  true - Akzeptiert die Lizenz. Das Produkt wird installiert.
#  false - Die Lizenz wird nicht akzeptiert. Es findet keine Installation statt.
#
# Wenn keine Installation stattfindet, wird dies in einer temporären
# Protokolldatei im temporären Verzeichnis des Benutzers aufgezeichnet.
#
# Wenn Sie die Option silentInstallLicenseAcceptance in dieser Antwortdatei auf
# "true" setzen, geben Sie an, dass Sie die internationalen Nutzungsbedingungen
# für Programmpakete der IBM unter
# CD_ROOT\XD\wxs.primary.pak\repository\legal.xs\license.xs
# gelesen haben und diese akzeptieren. Falls Sie diesen Bedingungen nicht
# zustimmen, dürfen Sie diesen Wert nicht ändern und das Programm weder
# herunterladen, noch installieren, kopieren, aufrufen oder verwenden. Geben
# Sie das Programm und den Berechtigungsnachweis umgehend an den Verkäufer
# zurück, um sich den Kaufbetrag erstatten zu lassen.
#
```

```
-OPT silentInstallLicenseAcceptance="false"


################################################################################
# Nicht blockierende Prüfung der Voraussetzungen
#
# Wenn Sie die nicht blockierende Prüfung der Voraussetzungen inaktivieren möchten,
# müssen Sie das Kommentarzeichen aus der folgenden Anweisungszeile entfernen.
# Mit dieser Option wird das Installationsprogramm angewiesen, die Installation
# fortzusetzen und die Warnungen zu protokollieren, selbst wenn die
# Prüfung der Voraussetzungen scheitert.
#
#-OPT disableNonBlockingPrereqChecking="true"


################################################################################
#
# Installationspfad:
#
# Dies ist das Installationsverzeichnis für das Produkt. Geben Sie ein gültiges
# Verzeichnis für die Installation des Produkts an. Falls der Verzeichnisname
# Leerzeichen enthält, setzen Sie ihn wie im folgenden Windows-Beispiel in
# Anführungszeichen. Installationsverzeichnisse mit Leerzeichen werden nur unter
# Windows-Betriebssystemen unterstützt. Die maximale Pfadlänge für Windows
# ist 60 Zeichen.
#
# Nachfolgend sind die Standardinstallationspfade für alle unterstützten
# Betriebssysteme aufgelistet, wenn Sie die Installation als Root durchführen.
# In dieser Antwortdatei wird standardmäßig das Installationsverzeichnis für
# Windows verwendet. Falls Sie das Standardinstallationsverzeichnis für ein
# anderes Betriebssystem verwenden möchten, entfernen Sie das Kommentarzeichen
# ('#') vor dem entsprechenden Verzeichnis, und fügen Sie vor dem Eintrag für
# das Windows-Betriebssystem ein Kommentarzeichen ('#') hinzu.
#
# Der Installationspfad wird verwendet, um festzustellen, ob WebSphere eXtreme
# Scale als eigenständige Implementierung installiert oder mit einer vorhandenen
# Installation von WebSphere Application Server integriert werden muss.
#
# Wenn der angegebene Pfad eine vorhandene Installation von WebSphere Application
# Server oder WebSphere Network Deployment enthält, wird eXtreme Scale mit dem
# vorhandenen Produkt integriert. Ist der angegebene Pfad neu oder leer, wird
# WebSphere eXtreme Scale als eigenständige Implementierung installiert.
#
# Anmerkung: Wenn der angegebene Installationspfad eine frühere
# Installation von WebSphere eXtreme Scale, WebSphere eXtended
# Deployment DataGrid oder ObjectGrid enthält, schlägt die
# Installation fehl:
#
# Standardinstallationspfad für AIX:
#
#  -OPT installLocation="/usr/IBM/WebSphere/eXtremeScale"
#
# Standardinstallationspfad für HP-UX, Solaris oder Linux:
#
#  -OPT installLocation="/opt/IBM/WebSphere/eXtremeScale"
#
#
# Standardinstallationspfad für Windows:
#
-OPT installLocation="C:\Program Files\IBM\WebSphere\eXtremeScale"

#
# Wenn Sie die Installation unter einer Benutzer-ID ohne Root- (UNIX)
# bzw. Administratorrechte (Windows) durchführen, werden die folgenden
# Standardinstallationspfade empfohlen.
# Stellen Sie sicher, dass Sie Schreibzugriff auf den ausgewählten
# Installationspfad haben.
```

```
#
# Standardinstallationspfad für AIX:
#
#  -OPT installLocation="<Ausgangsverzeichnis_des_Benutzers>/IBM/WebSphere/eXtremeScale"
#
# Standardinstallationspfad für HP-UX, Solaris oder Linux:
#
#  -OPT installLocation="<Ausgangsverzeichnis_des_Benutzers>/IBM/WebSphere/eXtremeScale"
#
# Standardinstallationspfad für Windows:
#
# -OPT installLocation="C:\IBM\WebSphere\eXtremeScale"


###############################################################################
#  Installation von Zusatzfunktionen
#
# Geben Sie an, welche Zusatzfunktionen Sie installieren möchten, indem Sie jede
# gewünschte Funktion auf "true" setzen. Setzen Sie alle Zusatzfunktionen, die
# Sie nicht installieren möchten, auf "false".
#
# Die Optionen selectServer, selectClient, selectPF und selectXSStreamQuery sind
# nur gültig, wenn die Option installLocation zuvor eine Installation von
# WebSphere Application Server enthält. Die Optionen werden in einer
# eigenständigen Installation von WebSphere eXtreme Scale ignoriert.
#
# In der eigenständigen Installation von WebSphere eXtreme Scale werden der
# eXtreme-Scale-Server und der eXtreme-Scale-Client automatisch installiert.
# Die Zusatzeinrichtungen für die eigenständige Installation von
# eXtreme Scale sind selectXSConsoleOther und selectXSStreamQueryOther.
#
# Wenn diese Option ausgewählt ist, werden die Komponenten installiert, die
# erforderlich sind, um Server von WebSphere eXtreme Scale und den dynamischen
# Cacheserviceprovider von eXtreme Scale auszuführen. Wenn diese Option
# ausgewählt wird, muss auch WebSphere eXtreme Scale Client ausgewählt werden,
# indem das Kommentarzeichen aus der Zeile mit der entsprechenden Option
# entfernt und die Option auf "true" gesetzt wird.
# Andernfalls schlägt die unbeaufsichtigte Installation fehl.
#
-OPT selectServer="true"

#
# Wenn diese Option ausgewählt ist, werden die Komponenten installiert, die
# erforderlich sind, um die Clientanwendungen von WebSphere eXtreme Scale
# auszuführen. Wenn die Serveroption ausgewählt wurde, muss diese Option
# ebenfalls ausgewählt werden, indem das Kommentarzeichen entfernt und die
# Option auf "true" gesetzt wird. Andernfalls schlägt die unbeaufsichtigte
# Installation fehl.
#
-OPT selectClient="true"

#
# Wenn diese Option ausgewählt ist, werden die Komponenten installiert, die
# erforderlich sind, um die Konsole von WebSphere eXtreme Scale auszuführen.
# Wenn diese Option ausgewählt wird, muss der zuvor angegebene Installationspfad
# ein neues oder leeres Verzeichnis sein, da die Konsoloption nur für die
# eigenständige Implementierung von WebSphere eXtreme Scale gültig ist. Zum
# Installieren dieser Zusatzeinrichtung muss das Kommentarzeichen aus der
# folgenden Optionszeile entfernt und die Option auf "true" gesetzt werden.
#-OPT selectXSConsoleOther="false"

#
# Wenn die folgenden Optionen ausgewählt werden, werden veraltete Funktionen
# installiert.
#
# Diese Option wählt WebSphere Partition Facility zur Installation aus.
# Diese Funktion ist veraltet. Zum Installieren dieser Zusatzeinrichtung
```

```
# muss das Kommentarzeichen aus der folgenden Optionszeile entfernt und
# die Option auf "true" gesetzt werden.
#
#-OPT selectPF="false"

#
# Diese Option wählt WebSphere eXtreme Scale StreamQuery for WAS zur
# Installation aus. Diese Funktion ist veraltet. Zum Installieren
# dieser Zusatzeinrichtung muss das Kommentarzeichen aus der folgenden
# Optionszeile entfernt und die Option auf "true" gesetzt werden.
# Wenn diese Option ausgewählt wird, muss auch WebSphere eXtreme Scale Client
# ausgewählt werden, indem das Kommentarzeichen aus der entsprechenden
# Optionszeile entfernt und die Option auf "true" gesetzt wird.
# Andernfalls schlägt die unbeaufsichtigte Installation fehl.
#
#-OPT selectXSStreamQuery="false"

#
# Diese Option wählt WebSphere eXtreme Scale StreamQuery for J2SE zur
# Installation aus. Diese Funktion ist veraltet. Zum Installieren
# dieser Zusatzeinrichtung muss das Kommentarzeichen aus der folgenden
# Optionszeile entfernt und die Option auf "true" gesetzt werden.
# Wenn diese Option ausgewählt wird, muss auch WebSphere eXtreme Scale Client
# ausgewählt werden, indem das Kommentarzeichen aus der entsprechenden
# Optionszeile entfernt und die Option auf "true" gesetzt wird.
# Andernfalls schlägt die unbeaufsichtigte Installation fehl.
#
#-OPT selectXSStreamQueryOther="false"


###############################################################################
#  Profilliste für Erweiterung
#
# Geben Sie an, welches der vorhandenen Profile Sie erweitern möchten, oder
# setzen Sie diese Zeile auf Kommentar, wenn alle vorhandenen Profile, die
# während der Installation erkannt werden, erweitert werden sollen.
#
# Wenn Sie mehrere Profile angeben möchten, verwenden Sie zwischen den
# einzelnen Namen ein Komma, z. B. "AppSrv01,Dmgr01,Custom01". Die Liste
# darf keine Leerzeichen enthalten.
#
-OPT profileAugmentList=""


###############################################################################
# Trace-Steuerung
#
# Das Format der Trace-Ausgabe kann über die folgende Option gesteuert werden:
# -OPT traceFormat=ALL
#
# Die Formatoptionen sind 'text' und 'XML'. Standardmäßig werden beide Formate
# in zwei unterschiedlichen Trace-Dateien erzeugt.
#
# Wenn nur ein Format erforderlich ist, können Sie mit der Option traceFormat das
# gewünschte Format wie folgt angeben:
#
# Gültige Werte:
#
#  text - Die Zeilen in der Trace-Datei haben zur besseren Lesbarkeit
#         ein reines Textformat.
#  XML  - Die Zeilen in der Trace-Datei haben das Standard-XML-Format für
#         Java-Protokollierung, das mit jedem Text- oder XML-Editor und mit
#         dem Tool Chainsaw von Apache
#         (http://logging.apache.org/log4j/docs/chainsaw.html) angezeigt
#         werden kann.
#
# Wie viele Trace-Informationen erfasst werden sollen, kann mit der
```

```
# folgenden Option gesteuert werden:
# -OPT traceLevel=INFO
#
# Gültige Werte:
#
# Stufe  Numer. Wert  Beschreibung
# -------  ---------  ------------------------------------------------------
# OFF       0         Es wird keine Trace-Datei erzeugt.
# SEVERE    1         Es werden nur schwerwiegende Fehler in der Trace-Datei
#                     ausgegeben.
# WARNING   2         Der Trace-Datei werden Nachrichten zu nicht schwerwiegenden
#                     Ausnahmen und Warnungen hinzugefügt.
# INFO      3         Der Trace-Datei werden Informationsnachrichten hinzugefügt.
#                     (Dies ist die Standard-Trace-Stufe.)
# CONFIG    4         Der Trace-Datei werden konfigurationsbezogene Nachrichten
#                     hinzugefügt.
# FINE      5         Es wird ein Trace für die Methodenaufrufe für allgemein
#                     zugängliche Methoden durchgeführt.
# FINER     6         Es wird ein Trace für die Methodenaufrufe für nicht
#                     allgemein zugängliche Methoden durchgeführt. Eine Ausnahme
#                     bilden Getter und Setter.
# FINEST    7         Es wird ein Trace für alle Methodenaufrufe durchgeführt.
#                     Der Trace für den Eintritt in die und den Austritt aus
#                     der Methode enthält die Parameter und den Rückgabewert.
```

Antwortdatei für die Installation von WebSphere eXtreme Scale Client

```
################################################################################
#
# InstallShield-Optionsdatei für IBM WebSphere eXtreme Scale Client V7.1.0
#
# Name des Assistenten: Install
# Quelle des Assistenten: setup.jar
#
# Diese Datei kann verwendet werden, um den Assistenten "Install" mit den im
# Folgenden angegebenen Optionen zu konfigurieren, wenn der Assistent mit der
# Befehlszeilenoption "-options" ausgeführt wird. Die Dokumentation zu jeder
# Einstellung enthält Informationen zum Ändern des Einstellungswerts.
# Schließen Sie alle Werte in Anführungszeichen ein.
#
# Eine gängige Verwendung für eine Optionsdatei ist die Ausführung des
# Assistenten im unbeaufsichtigten Modus. Auf diese Weise kann der Autor
# der Optionsdatei Assistenteneinstellungen festlegen, ohne den Assistenten
# im grafischen Modus oder im Konsolmodus ausführen zu müssen. Wenn Sie
# diese Optionsdatei für die Ausführung im unbeaufsichtigten Modus verwenden
# möchten, verwenden Sie die folgenden Befehlszeilenargumente bei der
# Ausführung des Assistenten:
#
#    -options "D:\installImage\WXS_Client\wxssetup.response" -silent
#
# Sie müssen den vollständig qualifizierten Namen der Antwortdatei angeben.
#
################################################################################

################################################################################
#
# Lizenz akzeptieren
#
# Gültige Werte:
#  true - Akzeptiert die Lizenz. Das Produkt wird installiert.
#  false - Die Lizenz wird nicht akzeptiert. Es findet keine Installation statt.
#
# Wenn keine Installation stattfindet, wird dies in einer temporären Protokolldatei
# im temporären Verzeichnis des Benutzers aufgezeichnet.
#
# Wenn Sie die Option silentInstallLicenseAcceptance in dieser Antwortdatei auf
# "true" setzen, geben Sie an, dass Sie die internationalen Nutzungsbedingungen
```

```
# für Programmpakete der IBM unter
# CD_ROOT\WXS_Cleint\wxs.client.primary.pak\repository\legal.xs.client\license.xs
# gelesen haben und diese akzeptieren. Falls Sie diesen Bedingungen nicht zustimmen,
# dürfen Sie diesen Wert nicht ändern und das Programm weder herunterladen,
# noch installieren, kopieren, aufrufen oder verwenden. Geben Sie das Programm
# und den Berechtigungsnachweis umgehend an den Verkäufer zurück, um sich den
# Kaufbetrag erstatten zu lassen.
#
-OPT silentInstallLicenseAcceptance="false"


################################################################################
# Nicht blockierende Prüfung der Voraussetzungen
#
# Wenn Sie die nicht blockierende Prüfung der Voraussetzungen inaktivieren möchten,
# müssen Sie das Kommentarzeichen aus der folgenden Anweisungszeile entfernen.
# Mit dieser Option wird das Installationsprogramm angewiesen, die Installation
# fortzusetzen und die Warnungen zu protokollieren, selbst wenn die
# Prüfung der Voraussetzungen scheitert.
#
#-OPT disableNonBlockingPrereqChecking="true"


################################################################################
#
# Installationspfad:
#
# Dies ist das Installationsverzeichnis für das Produkt. Geben Sie ein gültiges
# Verzeichnis für die Installation des Produkts an. Falls der Verzeichnisname
# Leerzeichen enthält, setzen Sie ihn wie im folgenden Windows-Beispiel in
# Anführungszeichen. Installationsverzeichnisse mit Leerzeichen werden nur unter
# Windows-Betriebssystemen unterstützt. Die maximale Pfadlänge für Windows
# ist 60 Zeichen.
#
# Nachfolgend sind die Standardinstallationspfade für alle unterstützten
# Betriebssysteme aufgelistet, wenn Sie die Installation als Root durchführen.
# In dieser Antwortdatei wird standardmäßig das Installationsverzeichnis für
# Windows verwendet. Falls Sie das Standardinstallationsverzeichnis für ein
# anderes Betriebssystem verwenden möchten, entfernen Sie das Kommentarzeichen
# ('#') vor dem entsprechenden Verzeichnis, und fügen Sie vor dem Eintrag für
# das Windows-Betriebssystem ein Kommentarzeichen ('#') hinzu.
#
# Der Installationspfad wird verwendet, um festzustellen, ob WebSphere eXtreme Scale
# als eigenständige Implementierung installiert oder mit einer vorhandenen
# Installation von WebSphere Application Server integriert werden muss.
#
# Wenn der angegebene Pfad eine vorhandene Installation von WebSphere Application Server
# oder WebSphere Network Deployment enthält, wird eXtreme Scale mit dem vorhanden
# Produkt integriert. Ist der angegebene Pfad neu oder leer, wird WebSphere
# eXtreme Scale als eigenständige Implementierung installiert.
#
# Anmerkung: Wenn der angegebene Installationspfad eine frühere
# Installation von WebSphere eXtreme Scale, WebSphere eXtended
# Deployment DataGrid oder ObjectGrid enthält, schlägt die
# Installation fehl:
#
# Standardinstallationspfad für AIX:
#
#  -OPT installLocation="/usr/IBM/WebSphere/eXtremeScale"
#
# Standardinstallationspfad für HP-UX, Solaris oder Linux:
#
#  -OPT installLocation="/opt/IBM/WebSphere/eXtremeScale"
#
#
# Standardinstallationspfad für Windows:
#
```

```
-OPT installLocation="C:\Program Files\IBM\WebSphere\eXtremeScale"

#
# Wenn Sie die Installation unter einer Benutzer-ID ohne Root- (UNIX)
# bzw. Administratorrechte (Windows) durchführen, werden die folgenden
# Standardinstallationspfade empfohlen.
# Stellen Sie sicher, dass Sie Schreibzugriff auf den ausgewählten
# Installationspfad haben.
#
# Standardinstallationspfad für AIX:
#
#  -OPT installLocation="<Ausgangsverzeichnis_des_Benutzers>/IBM/WebSphere/eXtremeScale"
#
# Standardinstallationspfad für HP-UX, Solaris oder Linux:
#
#  -OPT installLocation="<Ausgangsverzeichnis_des_Benutzers>/IBM/WebSphere/eXtremeScale"
#
# Standardinstallationspfad für Windows:
#
# -OPT installLocation="C:\IBM\WebSphere\eXtremeScale"


##############################################################################
#  Profilliste für Erweiterung
#
# Geben Sie an, welches der vorhandenen Profile Sie erweitern möchten, oder
# setzen Sie diese Zeile auf Kommentar, wenn alle vorhandenen Profile, die
# während der Installation erkannt werden, erweitert werden sollen.
#
# Wenn Sie mehrere Profile angeben möchten, verwenden Sie zwischen den
# einzelnen Namen ein Komma, z. B. "AppSrv01,Dmgr01,Custom01". Die Liste
# darf keine Leerzeichen enthalten.
#
-OPT profileAugmentList=""


##############################################################################
# Trace-Steuerung
#
# Das Format der Trace-Ausgabe kann über die folgende Option gesteuert werden:
# -OPT traceFormat=ALL
#
# Die Formatoptionen sind 'text' und 'XML'. Standardmäßig werden beide Formate
# in zwei unterschiedlichen Trace-Dateien erzeugt.
#
# Wenn nur ein Format erforderlich ist, können Sie mit der Option traceFormat das
# gewünschte Format wie folgt angeben:
#
# Gültige Werte:
#
#  text - Die Zeilen in der Trace-Datei haben zur besseren Lesbarkeit
#         ein reines Textformat.
#  XML  - Die Zeilen in der Trace-Datei haben das Standard-XML-Format für
#         Java-Protokollierung, das mit jedem Text- oder XML-Editor und mit
#         dem Tool Chainsaw von Apache
#         (http://logging.apache.org/log4j/docs/chainsaw.html) angezeigt
#         werden kann.
#
# Wie viele Trace-Informationen erfasst werden sollen, kann mit der
# folgenden Option gesteuert werden:
# -OPT traceLevel=INFO
#
# Gültige Werte:
#
# Stufe  Numer. Wert  Beschreibung
# -------  ---------  ---------------------------------------------------------
# OFF       0         Es wird keine Trace-Datei erzeugt.
```

```
# SEVERE    1         Es werden nur schwerwiegende Fehler in der Trace-Datei
#                     ausgegeben.
# WARNING   2         Der Trace-Datei werden Nachrichten zu nicht schwerwiegenden
#                     Ausnahmen und Warnungen hinzugefügt.
# INFO      3         Der Trace-Datei werden Informationsnachrichten hinzugefügt.
#                     (Dies ist die Standard-Trace-Stufe.)
# CONFIG    4         Der Trace-Datei werden konfigurationsbezogene Nachrichten
#                     hinzugefügt.
# FINE      5         Es wird ein Trace für die Methodenaufrufe für allgemein
#                     zugängliche Methoden durchgeführt.
# FINER     6         Es wird ein Trace für die Methodenaufrufe für nicht
#                     allgemein zugängliche Methoden durchgeführt. Eine Ausnahme
#                     bilden Getter und Setter.
# FINEST    7         Es wird ein Trace für alle Methodenaufrufe durchgeführt.
#                     Der Trace für den Eintritt in die und den Austritt aus
#                     der Methode enthält die Parameter und den Rückgabewert.
```

Profile für WebSphere eXtreme Scale erstellen und erweitern

Nachdem Sie das Produkt installiert haben, erstellen Sie eindeutige Typen von Profilen und erweitern vorhandene Profile für WebSphere eXtreme Scale.

Vorbereitende Schritte

Installieren Sie WebSphere eXtreme Scale. Weitere Informationen finden Sie im Abschnitt „WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client mit WebSphere Application Server integrieren” auf Seite 23.

Die Erweiterung von Profilen für WebSphere eXtreme Scale ist optional, aber in den folgenden Einsatzszenarien erforderlich:

- Automatisches Starten eines Katalogservice oder -containers in einem Prozess von WebSphere Application Server. Wenn die Serverprofile nicht erweitert werden, können Server nur über das Programm mit der API "ServerFactory" oder als gesonderte Prozesse mit Hilfe von Scripts gestartet werden.
- Verwendung von Performance Monitoring Infrastructure (PMI) für die Überwachung der eXtreme-Scale-Metriken.
- Anzeige der Version von WebSphere eXtreme Scale in der Administrationskonsole von WebSphere Application Server.

Informationen zu diesem Vorgang

Ausführung in WebSphere Application Server Version 6.0.2

Wenn Ihre Umgebung WebSphere Application Server Version 6.0.2 enthält, verwenden Sie den Befehl "wasprofile", um, wie im folgenden Beispiel gezeigt, Profile für WebSphere eXtreme Scale zu erstellen oder zu erweitern.

```
Installationsstammverzeichnis/bin/wasprofile.sh|bat -augment -profileName dmgr_01
-templatePath "C:/ProgramFiles/IBM/WebSphere/AppServer/profileTemplates/xs_augment/dmgr"
```

Weitere Informationen finden Sie im Artikel "Befehl wasprofile" im Information Center von WebSphere Application Server.

Ausführung in WebSphere Application Server Version 6.1 oder Version 7.0

Wenn Ihre Umgebung WebSphere Application Server Version 6.1 oder Version 7.0 enthält, können Sie das PMT-Plug-in (Profile Management Tool) oder den Befehl "manageprofiles" verwenden, um Profile zu erstellen und zu erweitern.

Nächste Schritte

Abhängig davon, welche Aufgabe Sie ausführen möchten, starten Sie dafür die Konsole "Erste Schritte", um Unterstützung bei der Konfiguration und beim Test Ihrer Produktumgebung zu erhalten. Die Konsole "Erste Schritte" befindet sich im Verzeichnis <*Installationsstammverzeichnis*>\firststeps\wxs\firststeps.bat. Sie können weitere Profile erstellen oder erweitern, indem Sie die vorherigen Tasks wiederholen.

Grafische Benutzerschnittstelle für die Erstellung von Profilen verwenden

Verwenden Sie die grafische Benutzerschnittstelle (GUI), die mit dem PMT-Plug-in (Profile Management Tool) bereitgestellt wird, um Profile für WebSphere eXtreme Scale zu erstellen. Ein Profil besteht aus einer Gruppe von Dateien, die die Laufzeitumgebung definieren.

Vorbereitende Schritte

Anmerkung: Wenn Sie WebSphere Application Server Version 6.0.2 oder WebSphere Application Server Network Deployment Version 6.0.2 ausführen, müssen Sie den Befehl "wasprofile" verwenden, um, wie im folgenden Beispiel gezeigt, ein Profil für WebSphere eXtreme Scale zu erstellen oder zu erweitern:

```
Installationsstammverzeichnis/bin/wasprofile.sh|bat -create -profileName dmgr_01
-templatePath "C:/ProgramFiles/IBM/WebSphere/AppServer/profileTemplates/xs_augment/dmgr"
```

Weitere Informationen finden Sie im Artikel "Befehl 'wasprofile'" im Information Center von WebSphere Application Server Version 6.0.

Informationen zu diesem Vorgang

Damit Sie die Produktfeatures nutzen können, aktiviert das PMT-Plug-in (Profile Management Tool) die GUI, um Sie bei der Konfiguration von Profilen, z. B. Profile von WebSphere Application Server, Deployment-Manager-Profilen, Zellenprofilen oder angepassten Profilen, zu unterstützen.

Vorgehensweise

Verwenden Sie die GUI von Profile Management Tool, um Profile zu erstellen. Wählen Sie eine der folgenden Optionen aus, um den Assistenten zu starten:

- Wählen Sie in der Konsole "Erste Schritte" die Option **Profile Management Tool** aus.
- Rufen Sie Profile Management Tool über das Menü **Start** auf.
- Führen Sie das Script "./pmt.sh|bat" im Verzeichnis *Installationsstammverzeichnis*/bin/ProfileManagement aus.

Nächste Schritte

Sie können weitere Profile erstellen oder vorhandene Profile erweitern. Zum erneuten Starten von Profile Management Tool führen Sie den Befehl "./pmt.sh|bat" im Verzeichnis *Installationsstammverzeichnis*/bin/ProfileManagement aus, oder wählen Sie **Profile Management Tool** in der Konsole "Erste Schritte" aus.

Starten Sie einen Katalogservice, starten Sie Container, und konfigurieren Sie TCP-Ports in Ihrer Umgebung mit WebSphere Application Server. Weitere Informationen finden Sie unter „WebSphere eXtreme Scale mit WebSphere Application Server verwalten” auf Seite 348.

Grafische Benutzerschnittstelle für die Erweiterung von Profilen verwenden

Nach der Installation des Produkts können Sie ein vorhandenes Profil erweitern, um es mit WebSphere eXtreme Scale kompatibel zu machen.

Vorbereitende Schritte

Anmerkung: Wenn Sie WebSphere Application Server Version 6.0.2 oder WebSphere Application Server Network Deployment Version 6.0.2 ausführen, müssen Sie den Befehl "wasprofile" verwenden, um, wie im folgenden Beispiel gezeigt, ein Profil für WebSphere eXtreme Scale zu erstellen oder zu erweitern:

```
Installationsstammverzeichnis/bin/wasprofile.sh|bat -augment -profileName dmgr_01
-templatePath "C:/ProgramFiles/IBM/WebSphere/AppServer/profileTemplates/xs_augment/dmgr"
```

Weitere Informationen finden Sie im Artikel "Befehl 'wasprofile'" im Information Center von WebSphere Application Server.

Informationen zu diesem Vorgang

Wenn Sie ein vorhandenes Profil erweitern, ändern Sie das Profil, indem Sie eine produktspezifische Erweiterungsschablone anwenden. Server von WebSphere eXtreme Scale werden beispielsweise nicht automatisch gestartet, wenn das Serverprofil nicht mit der Schablone "xs_augment" erweitert wurde.

- Erweitern Sie das Profil mit der Schablone "xs_augment", wenn Sie den eXtreme-Scale-Client oder den Client und den Server installiert haben.
- Erweitern Sie das Profil nur dann mit der Schablone "pf_augment", wenn Sie das Partitionierungsfeature installiert haben.
- Wenden Sie beide Schablonen an, wenn Ihre Umgebung den eXtreme-Scale-Client und das Partitionierungsfeature enthält.

Vorgehensweise

Verwenden Sie die GUI von Profile Management Tool, um Profile für eXtreme Scale zu erweitern. Wählen Sie eine der folgenden Optionen aus, um den Assistenten zu starten:

- Wählen Sie in der Konsole "Erste Schritte" die Option **Profile Management Tool** aus.
- Rufen Sie Profile Management Tool über das Menü **Start** auf.
- Führen Sie das Script "./pmt.sh|bat" im Verzeichnis *Installationsstammverzeichnis*/bin/ProfileManagement aus.

Nächste Schritte

Sie können weitere Profile erweitern. Zum erneuten Starten von Profile Management Tool führen Sie den Befehl "./pmt.sh|bat" im Verzeichnis *Installationsstammverzeichnis*/bin/ProfileManagement aus, oder wählen Sie **Profile Management Tool** in der Konsole "Erste Schritte" aus.

Starten Sie einen Katalogservice, starten Sie Container, und konfigurieren Sie TCP-Ports in Ihrer Umgebung mit WebSphere Application Server. Weitere Informationen hierzu finden Sie im Abschnitt „WebSphere eXtreme Scale mit WebSphere Application Server verwalten” auf Seite 348.

Befehl "manageprofiles"

Sie können das Dienstprogramm "manageprofiles" verwenden, um Profile mit der eXtreme-Scale-Schablone zu erstellen und vorhandene Anwendungsserverprofile mit den eXtreme-Scale-Erweiterungsschablonen zu erweitern bzw. deren Erweiterung aufzuheben. Zur Verwendung der Produktfeatures muss Ihre Umgebung mindestens ein Profil enthalten, das für das Produkt erweitert wurde.

- Bevor Sie Profile erstellen und erweitern können, müssen Sie eXtreme Scale installieren. Weitere Informationen finden Sie im Abschnitt „WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client mit WebSphere Application Server integrieren” auf Seite 23.
- Wenn Ihre Umgebung WebSphere Application Server Version 6.0.2 enthält, müssen Sie den Befehl "wasprofile" verwenden, um Profile für eXtreme Scale zu erstellen und zu erweitern, wie im folgenden Beispiel gezeigt wird:

  ```
  Installationsstammverzeichnis/bin/wasprofile.sh|bat -augment -profileName dmgr_01
  -templatePath "C:/ProgramFiles/IBM/WebSphere/AppServer/profileTemplates/xs_augment/dmgr"
  ```

  Weitere Informationen finden Sie im Artikel "Befehl wasprofile" im Information Center von WebSphere Application Server.

Zweck

Der Befehl "manageprofiles" erstellt die Laufzeitumgebung für einen Produktprozess in einem Satz von Dateien, einem so genannten Profil. Das Profil definiert die Laufzeitumgebung. Sie können mit dem Befehl "manageprofiles" die folgenden Aktionen ausführen:

- Deployment-Manager-Profil erstellen und erweitern
- Angepasstes Profil erstellen und erweitern
- Eigenständiges Anwendungsserverprofil erstellen und erweitern
- Zellenprofil erstellen und erweitern
- Erweiterung jedes Typs von Profil aufheben

Wenn Sie ein vorhandenes Profil erweitern, ändern Sie das Profil, indem Sie eine produktspezifische Erweiterungsschablone anwenden.

- Erweitern Sie das Profil mit der Schablone "xs_augment", wenn Sie den eXtreme-Scale-Client oder den Client und den Server installiert haben.
- Erweitern Sie das Profil mit der Schablone "pf_augment", wenn Sie nur das Partitionierungsfeature installiert haben.
- Wenden Sie beide Schablonen an, wenn Ihre Umgebung den eXtreme-Scale-Client und das Partitionierungsfeature enthält.

Position

Die Befehlsdatei befindet sich im Verzeichnis *Installationsstammverzeichnis*/bin.

Verwendung

Ausführliche Hilfe können Sie mit dem Parameter **-help** abrufen:

```
./manageprofiles.sh|bat -create -templatePath Installationsstammverzeichnis/profileTemplates/xs_augment/dmgr -help
```

In den folgenden Abschnitten wird jede Task, die Sie mit dem Befehl "manageprofiles" ausführen können, zusammen mit einer Liste der erforderlichen Parameter beschrieben. Einzelheiten zu den optionalen Parametern, die Sie für jede Task angeben können finden Sie im Artikel "Befehl manageprofiles" im Information Center von WebSphere Application Server.

Deployment-Manager-Profil erstellen

Sie können den Befehl "manageprofiles" verwenden, um ein Deployment-Manager-Profil zu erstellen. Der Deployment Manager verwaltet die Anwendungsserver, die in die Zelle eingebunden sind.

Parameter

-create
Erstellt ein Profil. (Erforderlich)

-templatePath *Schablonenpfad*
Gibt den Dateipfad der Schablone an. (Erforderlich)

Verwenden Sie das folgende Format:

```
-templatePath Installationsstammverzeichnis/profileTemplates/Schablonentyp/dmgr
```

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":

```
./manageprofile.sh|bat -create -templatePath
Installationsstammverzeichnis/profileTemplates/xs_augment/dmgr
```

- Mit der Schablone "pf_augment":

```
./manageprofile.sh|bat -create -templatePath
Installationsstammverzeichnis/profileTemplates/pf_augment/dmgr
```

Angepasstes Profil erstellen

Sie können den Befehl "manageprofiles" verwenden, um ein angepasstes Profil zu erstellen. Ein angepasstes Profil ist ein leerer Knoten, den Sie über den Deployment Manager anpassen, indem Sie Anwendungsserver, Cluster oder andere Java-Prozesse aufnehmen.

Parameter

-create
Erstellt ein Profil. (Erforderlich)

-templatePath *Schablonenpfad*
Gibt den Dateipfad der Schablone an. (Erforderlich)

Verwenden Sie das folgende Format:

```
-templatePath Installationsstammverzeichnis/profileTemplates/Schablonentyp/managed
```

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":

```
./manageprofile.sh|bat -create -templatePath Installationsstammverzeichnis/profileTemplates/xs_augment/managed
```

- Mit der Schablone "pf_augment":

```
./manageprofile.sh|bat -create -templatePath Installationsstammverzeichnis/profileTemplates/pf_augment/managed
```

Eigenständiges Anwendungsserverprofil erstellen

Sie können den Befehl "manageprofiles" verwenden, um ein eigenständiges Anwendungsserverprofil zu erstellen.

Parameter

-create
Erstellt ein Profil. (Erforderlich)

-templatePath *Schablonenpfad*
Gibt den Dateipfad der Schablone an. (Erforderlich)

Verwenden Sie das folgende Format:

```
-templatePath Installationsstammverzeichnis/profileTemplates/Schablonentyp/default
```

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":

```
./manageprofile.sh|bat -create -templatePath Installationsstammverzeichnis/profileTemplates/xs_augment/default
```

- Mit der Schablone "pf_augment":

```
./manageprofile.sh|bat -create -templatePath Installationsstammverzeichnis/profileTemplates/pf_augment/default
```

Zellenprofil erstellen

Sie können den Befehl "manageprofiles" verwenden, um ein Zellenprofil zu erstellen, das sich aus einem Deployment Manager und einem Anwendungsserver zusammensetzt.

Parameter

Geben Sie die folgenden Parameter in der Deployment-Manager-Schablone an:

-create
Erstellt ein Profil. (Erforderlich)

-templatePath *Schablonenpfad*
Gibt den Dateipfad der Schablone an. (Erforderlich)

Verwenden Sie das folgende Format:

```
-templatePath Installationsstammverzeichnis/profileTemplates/Schablonentyp/cell/dmgr
```

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Geben Sie die folgenden Parameter für die Anwendungsserverschablone an:

-create
Erstellt ein Profil. (Erforderlich)

-templatePath *Schablonenpfad*
Gibt den Dateipfad der Schablone an. (Erforderlich)

Verwenden Sie das folgende Format:

```
-templatePath Installationsstammverzeichnis/profileTemplates/Schablonentyp/cell/default
```

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":

```
./manageprofile.sh|bat -create -templatePath Installationsstammverzeichnis/profileTemplates/xs_augment/cell/dmgr
-nodeProfilePath Installationsstammverzeichnis/profiles/AppSrv01 -cellName cell01dmgr -nodeName node01dmgr
-appServerNodeName node01

./manageprofile.sh|bat -create -templatePath Installationsstammverzeichnis/profileTemplates/xs_augment/cell/default
```

```
-dmgrProfilePath Installationsstammverzeichnis/profiles/Dmgr01 -portsFile
Installationsstammverzeichnis/profiles/Dmgr01/properties/portdef.props -nodePortsFile
Installationsstammverzeichnis/profiles/Dmgr01/properties/nodeportdef.props -cellName cell01dmgr
-nodeName node01dmgr -appServerNodeName node01
```

- Mit der Schablone "pf_augment":

```
./manageprofile.sh|bat -create -templatePath Installationsstammverzeichnis/profileTemplates/pf_augment/cell/dmgr
-nodeProfilePath Installationsstammverzeichnis/profiles/AppSrv01 -cellName cell01dmgr -nodeName node01dmgr
-appServerNodeName node01

./manageprofile.sh|bat -create -templatePath Installationsstammverzeichnis/profileTemplates/pf_augment/cell/default
-dmgrProfilePath Installationsstammverzeichnis/profiles/Dmgr01 -portsFile
Installationsstammverzeichnis/profiles/Dmgr01/properties/portdef.props -nodePortsFile
Installationsstammverzeichnis/profiles/Dmgr01/properties/nodeportdef.props -cellName cell01dmgr
-nodeName node01dmgr -appServerNodeName node01
```

Deployment-Manager-Profil erweitern

Sie können den Befehl "manageprofiles" verwenden, um ein Deployment-Manager-Profil zu erweitern.

Parameter

-augment
Erweitert ein vorhandenes Profil. (Erforderlich)

-profileName
Gibt den Namen des Profils an. (Erforderlich)

-templatePath *Schablonenpfad*
Gibt den Pfad der Schablonendateien an, die sich im Installationsstammverzeichnis befinden. (Erforderlich)

Verwenden Sie das folgende Format:

```
-templatePath Installationsstammverzeichnis/profileTemplates/Schablonentyp/dmgr
```

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":

```
./manageprofile.sh|bat -augment -profileName profile01
-templatePath Installationsstammverzeichnis/profileTemplates/xs_augment/dmgr
```

- Mit der Schablone "pf_augment":

```
./manageprofile.sh|bat -augment -profileName profile01
-templatePath Installationsstammverzeichnis/profileTemplates/pf_augment/dmgr
```

Angepasstes Profil erweitern

Sie können den Befehl "manageprofiles" verwenden, um ein angepasstes Profil zu erweitern.

Parameter

-augment
Erweitert ein vorhandenes Profil. (Erforderlich)

-profileName
Gibt den Namen des Profils an. (Erforderlich)

-templatePath *Schablonenpfad*
Gibt den Pfad der Schablonendateien an, die sich im Installationsstammverzeichnis befinden. (Erforderlich)

Verwenden Sie das folgende Format:

```
-templatePath Installationsstammverzeichnis/profileTemplates/Schablonentyp/managed
```

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":
  `./manageprofile.sh|bat -augment -profileName profile01 -templatePath Installationsstammverzeichn`
- Mit der Schablone "pf_augment":
  `./manageprofile.sh|bat -augment -profileName profile01 -templatePath Installationsstammverzeichn`

Eigenständiges Anwendungsserverprofil erweitern

Sie können den Befehl "manageprofiles" verwenden, um ein eigenständiges Anwendungsserverprofil zu erweitern.

Parameter

-augment
Erweitert ein vorhandenes Profil. (Erforderlich)

-profileName
Gibt den Namen des Profils an. (Erforderlich)

-templatePath *Schablonenpfad*
Gibt den Pfad der Schablonendateien an, die sich im Installationsstammverzeichnis befinden. (Erforderlich)

Verwenden Sie das folgende Format:
`-templatePath Installationsstammverzeichnis/profileTemplates/Schablonentyp/default`

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":
  `./manageprofile.sh|bat -augment -profileName profile01 -templatePath Installationsstammverzeichn`
- Mit der Schablone "pf_augment":
  `./manageprofile.sh|bat -augment -profileName profile01 -templatePath Installationsstammverzeichn`

Zellenprofil erweitern

Sie können den Befehl "manageprofiles" verwenden, um ein Zellenprofil zu erweitern.

Parameter

Geben Sie die folgenden Parameter für das Deployment-Manager-Profil an:

-augment
Erweitert ein vorhandenes Profil. (Erforderlich)

-profileName
Gibt den Namen des Profils an. (Erforderlich)

-templatePath *Schablonenpfad*
Gibt den Pfad der Schablonendateien an, die sich im Installationsstammverzeichnis befinden. (Erforderlich)

Verwenden Sie das folgende Format:
`-templatePath Installationsstammverzeichnis/profileTemplates/Schablonentyp/cell/dmgr`

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Geben Sie die folgenden Parameter für das Anwendungsserverprofil an:

-augment
Erweitert ein vorhandenes Profil. (Erforderlich)

-profileName
Gibt den Namen des Profils an. (Erforderlich)

-templatePath *Schablonenpfad*
Gibt den Pfad der Schablonendateien an, die sich im Installationsstammverzeichnis befinden. (Erforderlich)

Verwenden Sie das folgende Format:

```
-templatePath Installationsstammverzeichnis/profileTemplates/Schablonentyp/cell/default
```

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":

```
./manageprofile.sh|bat -augment -profileName profile01 -templatePath Installationsstammverzeichnis/profileTemplates/xs_augment/cell/dmgr
./manageprofile.sh|bat -augment -profileName profile01 -templatePath Installationsstammverzeichnis/profileTemplates/xs_augment/cell/defau
```

- Mit der Schablone "pf_augment":

```
./manageprofile.sh|bat -augment -profileName profile01 -templatePath Installationsstammverzeichnis/profileTemplates/pf_augment/cell/dmgr
./manageprofile.sh|bat -augment -profileName profile01 -templatePath Installationsstammverzeichnis/profileTemplates/pf_augment/cell/defau
```

Erweiterung eines Profils aufheben

Wenn Sie die Erweiterung eines Profils aufheben möchten, geben Sie den Parameter **-ignoreStack** mit dem Parameter **-templatePath** und die erforderlichen Parameter **-unaugment** und **-profileName** an.

Parameter

-unaugment
Hebt die Erweiterung eines zuvor erweiterten Profils auf. (Erforderlich)

-profileName
Gibt den Namen des Profils an. Der Parameter wird standardmäßig verwendet, wenn keine Werte angegeben sind. (Erforderlich)

-templatePath *Schablonenpfad*
Gibt den Pfad der Schablonendateien an, die sich im Installationsstammverzeichnis befinden. (Optional)

Verwenden Sie das folgende Format:

```
-templatePath Installationsstammverzeichnis/profileTemplates/Schablonentyp/Profiltyp
```

Schablonentyp steht für `xs_augment` oder `pf_augment` und *Profiltyp* für einen der folgenden vier Profiltypen:

- `dmgr`: Deployment-Manager-Profil
- `managed`: Angepasstes Profil
- `default`: Eigenständiges Anwendungsserverprofil
- `cell`: Zellenprofil

-ignoreStack
Wird zusammen mit dem Parameter **-templatePath** verwendet, um die Erweiterung eines bestimmten erweiterten Profils aufzuheben. (Optional)

Beispiel

- Mit der Schablone "xs_augment":

  ```
  ./manageprofile.sh|bat -unaugment -profileName profile01 -ignoreStack
  -templatePath Installationsstammverzeichnis/profileTemplates/xs_augment/Profiltyp
  ```

- Mit der Schablone "pf_augment":

  ```
  ./manageprofile.sh|bat -unaugment -profileName profile01 -ignoreStack
  -templatePath Installationsstammverzeichnis/profileTemplates/pf_augment/Profiltyp
  ```

Profile ohne Root-Rechte

Sie können einem Benutzer ohne Root-Rechte Berechtigungen für Dateien und Verzeichnisse erteilen, so dass dieser ein Profil für das Produkt erstellen kann. Der Benutzer ohne Root-Rechte kann auch ein Profil erweitern, das von einem Root-Benutzer oder von ihm selbst erstellt wurde.

In einer Umgebung von WebSphere Application Server können Benutzer ohne Root- bzw. Administratorrechte nur Profile in ihren eigenen Umgebungen erstellen und verwenden. Im PMT-Plug-in (Profile Management Tool) sind eindeutige Namen und Portwerte für Benutzer ohne Root-Rechte inaktiviert. Ein Benutzer ohne Root-Rechte muss die Standardfeldwerte für Profilnamen, Knotennamen, Zellennamen und Portzuordnungen in Profile Management Tool ändern. Daher sollte den Benutzern ohne Root-Rechte ein Wertebereich für jedes dieser Felder zugewiesen werden. Sie können die Zuständigkeit für die Verwendung der richtigen Wertebereiche und die Wahrung der Integrität ihrer eigenen Definitionen an die Benutzer delegieren.

Der Begriff *Installationsverantwortlicher* bezieht sich sowohl auf Benutzer mit Root-Rechten als auch auf Benutzer ohne Root-Rechte. Als Installationsverantwortlicher können Sie Benutzern ohne Root-Rechte Berechtigungen zum Erstellen von Profilen und zum Erstellen ihrer eigenen Produktumgebungen erteilen. Beispielsweise könnte ein Benutzer ohne Root-Rechte eine Produktumgebung erstellen, um die Anwendungsimplementierung mit einem ihm bekannten Profil zu testen. Sie können die folgenden speziellen Tasks ausführen, um das Erstellen eines Profils ohne Root-Rechte zuzulassen:

- Ein Profil erstellen und einen Benutzer ohne Root-Rechte zum Eigner des Profilverzeichnisses machen, so dass dieser WebSphere Application Server für ein bestimmtes Profil starten kann.
- Einem Benutzer ohne Root-Rechte Schreibzugriff für die entsprechenden Dateien und Verzeichnisse erteilen, damit dieser das Profil erstellen kann. Wenn Sie diese Task ausführen, können Sie eine Gruppe von Benutzern erstellen, die zum Erstellen von Profilen berechtigt sind, oder einzelnen Benutzern die Berechtigung zum Erstellen von Profilen erteilen.
- Wartungspakete für das Produkt installieren, einschließlich der erforderlichen Services für vorhandene Profile, die Eigentum eines Benutzers ohne Root-Rechte sind. Als Installationsverantwortlicher sind Sie Eigner aller neuen Dateien, die vom Wartungspaket erstellt werden.

Weitere Informationen zum Erstellen von Profilen für Benutzer ohne Root-Rechte finden Sie unter Profile für Benutzer ohne Root-Rechte erstellen.

Als Installationsverantwortlicher können Sie die Berechtigung zum Erweitern von Profilen ebenfalls einem Benutzer ohne Root-Rechte erteilen. Beispielsweise kann ein Benutzer ohne Root-Rechte ein Profil erweitern, das von einem Installationsverantwortlichen oder von ihm selbst erstellt wurde. Befolgen Sie in WebSphere Application Server Network Deployment den Prozess zur Erweiterung von Profilen für Benutzer ohne Root-Rechte.

Wenn ein Benutzer ohne Root-Rechte jedoch ein Profil erweitert, das vom Installationsverantwortlichen erstellt wurde, muss der Benutzer ohne Root-Rechte die folgenden Dateien vor der Erweiterung nicht erstellen: Die folgenden Dateien werden während des Profilerstellungsprozesses erstellt:

- *Stammverzeichnis_des_Anwendungsservers*/logs/manageprofiles.xml
- *Stammverzeichnis_des_Anwendungsservers*/properties/fsdb.xml
- *Stammverzeichnis_des_Anwendungsservers*/properties/profileRegistry.xml

Wenn ein Benutzer ohne Root-Rechte ein von ihm selbst erstelltes Profil erweitert, muss er die Berechtigungen für die Dokumente ändern, die sich in den Profilschablonen von eXtreme Scale befinden.

Achtung: Sie können ein Profil ohne Root-Rechte (Administratorrechte) auch für WebSphere eXtreme Scale in einer eigenständigen Umgebung verwenden, d. h. einer Umgebung außerhalb von WebSphere Application Server. Sie müssen den Eigner des ObjectGrid-Verzeichnisses in das Profil ohne Root-Rechte ändern. Anschließend können Sie sich mit diesem Profil ohne Root-Rechte anmelden und wie mit einem Profil mit Root-Rechten (Administratorrechten) mit eXtreme Scale arbeiten.

WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client unbeaufsichtigt installieren

Verwenden Sie eine vollständig qualifizierte Antwortdatei, die Sie speziell für Ihre Anforderungen konfigurieren, oder übergeben Sie Parameter in der Befehlszeile, um WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client unbeaufsichtigt zu installieren.

Vorbereitende Schritte

- Stoppen Sie alle Prozesse, die in Ihrer Umgebung mit WebSphere Application Server oder WebSphere Application Server Network Deployment aktiv sind. Weitere Informationen zu den Befehlen "stopManager", "stopNode" und "stopServer" finden Sie unter "wsadmin-Scripting-Befehlszeilentools verwenden".

  Vorsicht:
  Stellen Sie sicher, dass alle aktiven Prozesse gestoppt sind. Wenn die aktiven Prozesse nicht gestoppt sind, wird die Installation fortgesetzt, was zu unvorhersehbaren Ereignissen führen und die Installation auf einigen Plattformen in einem unbestimmten Zustand hinterlassen kann.
- Stellen Sie sicher, dass das Zielinstallationsverzeichnis leer bzw. nicht vorhanden ist.

  Wichtig: Wenn eine frühere Version von WebSphere eXtreme Scale oder die Komponente "ObjectGrid" in dem Verzeichnis vorhanden ist, das Sie für die Installation von Version 7.1 angeben, wird das Produkt nicht installiert. Sie können beispielsweise bereits einen Ordner <WXS-Installationsstammverzeichnis>/ObjectGrid haben. In diesem Fall können Sie ein anderes Installationsverzeichnis angeben oder die Installation abbrechen. Anschließend deinstallieren Sie die frühere Installation und führen dann den Assistenten erneut aus.

Informationen zu diesem Vorgang

Bei einer unbeaufsichtigten Installation wird dasselbe Installationsprogramm verwendet, das auch bei der Installation über die grafische Benutzerschnittstelle verwendet wird. Bei der unbeaufsichtigten Installation wird jedoch keine Assistentenschnittstelle angezeigt. Stattdessen werden Ihre Antworten aus einer Datei gelesen,

die Sie anpassen, bzw. aus Parametern, die die in der Befehlszeile übergeben. Sehen Sie sich das Beispiel für die „Datei wxssetup.response.txt" auf Seite 36 an, das eine Beschreibung jeder Option enthält.

Vorgehensweise

1. Optional: Wenn Sie sich für die Installation von WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client über eine Antwortdatei entscheiden, müssen Sie zuerst die Datei wxssetup.response.txt anpassen.

   Hinweis: Sie müssen den vollständig qualifizierten Namen der Antwortdatei angeben. Wenn Sie den relativen Pfad angeben, scheitert die Installation ohne Ausgabe einer entsprechenden Fehlernachricht.

   a. Erstellen Sie eine Kopie der anzupassenden Antwortdatei.

      Für die vollständige Installation von WebSphere eXtreme Scale kopieren Sie die Antwortdatei von der Produkt-DVD auf Ihr Plattenlaufwerk.

      Für WebSphere eXtreme Scale Client dekomprimieren Sie die ZIP-Datei von WebSphere eXtreme Scale Client auf Ihrem Festplattenlaufwerk. Suchen Sie anschließend die Antwortdatei.

   b. Öffnen Sie die Antwortdatei mit einem Texteditor Ihrer Wahl, und bearbeiten Sie sie. Die vorherige Beispielantwortdatei enthält Details zur Angabe der einzelnen Parameter. Sie müssen die folgenden Parameter angeben:

      - Lizenzvereinbarung
      - Installationsverzeichnis

      Tipp: Wenn Sie WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client in einer Umgebung von WebSphere Application Server installieren, verwendet das Installationsprogramm das Installationsverzeichnis, um zu bestimmen, wo die vorhandene Instanz von WebSphere Application Server installiert ist. Wenn Sie die Installation auf einem Knoten durchführen, der mehrere Instanzen von WebSphere Application Server enthält, müssen Sie Ihren Standort eindeutig definieren.

   c. Führen Sie das folgende Script aus, um die Installation zu starten.

      Für die vollständige Installation von WebSphere eXtreme Scale:

      ```
      ./install.sh|bat -options C:/Laufwerkspfad/Antwortdatei.txt -silent
      ```

      Für die Installation von WebSphere eXtreme Scale Client:

      ```
      ./WXS_Client/install.sh|bat -options C:/Laufwerkspfad/Antwortdatei.txt -silent
      ```

      Sie können die Antwortdatei auch verwenden, wenn Sie eine Installation über die grafische Benutzerschnittstelle durchführen. Mit der Antwortdatei können Sie bei einer Installation über die grafische Benutzerschnittstelle Probleme beheben, die bei der unbeaufsichtigten Installation verborgen bleiben. Wenn Sie die Datei wxssetup.response für Installationen über die grafische Benutzerschnittstelle und für unbeaufsichtigte Installationen angeben, müssen Sie den vollständig qualifizierten Pfad verwenden. Führen Sie das folgende Script aus, um die Installation über die grafische Benutzerschnittstelle mit der Antwortdatei durchzuführen:

      - Linux UNIX `<Ausgangsverzeichnis_der_Installation>/install.sh -options <vollständig_qualifizierter_Installationspfad>/wxssetup.response`
      - Windows `<Ausgangsverzeichnis_der_Installation>\install.exe -options c:\<vollständig_qualifizierter_Installationspfad>\wxssetup.response`

2. Optional: Wenn Sie sich für die Installation von eXtreme Scale durch Übergabe bestimmter Parameter in der Befehlszeile entscheiden, führen Sie das folgende Script aus, um die Installation zu starten:

   Für die vollständige Installation von WebSphere eXtreme Scale:

   ```
   ./install.sh|bat -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=Installationsposition
   ```

   Für die Installation von WebSphere eXtreme Scale Client:

   ```
   ./WXS_Client/install.sh|bat -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=Installationsposition
   ```

Installationsparameter

Geben Sie Parameter in der Befehlszeile an, um Ihre Produktinstallation anzupassen und zu konfigurieren.

Anmerkung: Sie müssen den vollständig qualifizierten Namen der Antwortdatei angeben. Wenn Sie den relativen Pfad angeben, scheitert die Installation ohne Ausgabe einer entsprechenden Fehlernachricht.

Parameter

Sie können die folgenden Parameter bei einer Installation des Produkts über die Befehlszeile oder eine Optionsdatei übergeben:

-silent
Unterdrückt die grafische Benutzerschnittstelle (GUI). Geben Sie den Parameter **-options** an, um anzuzeigen, dass das Installationsprogramm die Installation auf der Basis einer angepassten Optionsdatei durchführen soll. Wenn Sie den Parameter **-options** nicht angeben, werden die Standardwerte verwendet.

Beispielsyntax

```
./install.sh|bat -silent -options Optionsdatei.txt
```

-options *Pfadname/Dateiname*
Gibt eine Optionsdatei an, die das Installationsprogramm für die Durchführung einer unbeaufsichtigten Installation verwenden soll. Angaben in der Befehlszeile haben Vorrang.

Beispielsyntax

```
./install.sh|bat -options c:/Pfadname/Optionsdatei.txt
```

-log # !file_name @*Ereignistyp*
Generiert eine Installationsprotokolldatei, in der die folgenden Ereignistypen protokolliert werden:

- `err`
- `wrn`
- `msg1`
- `msg2`
- `dbg`
- `ALL`

Beispielsyntax

```
./install.sh|bat -log # !c:/temp/logfiles.txt @ALL
```

-is:log *Pfadname/Dateiname*
Erstellt eine Protokolldatei, die die Suchoperationen des Installationsprogramms in der JVM beim Versuch, die GUI zu starten, enthält. Die Protokolldatei wird nur erstellt, wenn Sie dieses angeben.

Beispielsyntax

```
./install.sh|bat -is:log c:/logs/javalog.txt
```

-is:javaconsole
Während des Installationsprozesses wird ein Konsolfenster angezeigt.

Beispielsyntax

```
./install.sh|bat -is:javaconsole
```

-is:silent
Unterdrückt das Java-Initialisierungsfenster, das beim Start des Installationsprogramms angezeigt wird.

Beispielsyntax

```
./install.sh|bat -is:silent
```

-is:tempdir *Pfadname*
Gibt das temporäre Verzeichnis an, das vom Installationsprogramm während der Installation verwendet wird.

Beispielsyntax

```
./install.sh|bat -is:tempdir c:/temp
```

Update Installer zum Installieren von Wartungspaketen verwenden

Verwenden Sie IBM Update Installer, um Ihre Umgebung von WebSphere eXtreme Scale mit verschiedenen Typen von Wartungspaketen, z. B. vorläufigen Fixes, Fixpacks und Refresh-Packs, zu aktualisieren.

Informationen zu diesem Vorgang

Verwenden Sie IBM Update Installer, um verschiedene Typen von Wartungspaketen für WebSphere eXtreme Scale zu installieren und anzuwenden. Da Update Installer in regelmäßigen Abständen aktualisiert wird, müssen Sie die aktuellste Version des Tools verwenden.

Vorgehensweise

1. Stoppen Sie alle Prozesse, die in Ihrer Umgebung aktiv sind.
   - Weitere Einzelheiten zum Stoppen aller Prozesse, die in der eigenständigen Umgebung von eXtreme Scale ausgeführt werden, finden Sie im Abschnitt „Eigenständige eXtreme-Scale-Server stoppen" auf Seite 344.
   - Weitere Informationen zum Stoppen aller Prozesse, die in der Umgebung von WebSphere Application Server ausgeführt werden, finden Sie unter "Befehlszeilendienstprogramme".
2. Laden Sie die aktuelle Version von Update Installer herunter. Weitere Informationen finden Sie auf der Webseite "Recommended fixes".
3. Installieren Sie Update Installer. Weitere Informationen finden Sie im Artikel "Update Installer für WebSphere Software" im Information Center von WebSphere Application Server.
4. Laden Sie die Wartungspakete, die Sie installieren möchten, in das Verzeichnis *UPDI-Stammverzeichnis*/maintenance herunter. Weitere Informationen finden Sie auf der Unterstützungssite.
5. Verwenden Sie Update Installer, um den vorläufigen Fix, das Fixpack oder das Refresh-Pack zu installieren. Sie können das Wartungspaket installieren, indem Sie die grafische Benutzerschnittstelle verwenden oder Update Installer im unbeaufsichtigten Modus ausführen.

Führen Sie den folgenden Befehl im Verzeichnis *UPDI-Stammverzeichnis* aus, um die grafische Benutzerschnittstelle zu starten:

- Linux UNIX `update.sh`
- Windows `update.bat`

Führen Sie den folgenden Befehl im Verzeichnis *UPDI-Stammverzeichnis* aus, um Update Installer im unbeaufsichtigten Modus auszuführen:

- Linux UNIX `./update.sh -silent -options responsefile/Dateiname`
- Windows `update.bat -silent -options responsefile\Dateiname`

Wenn der Installationsprozess fehlschlägt, sehen Sie sich die temporäre Protokolldatei im Verzeichnis *UDPI-Stammverzeichnis*`/logs/update/tmp` an. Update Installer erstellt das Verzeichnis *Installationsstammverzeichnis*`/logs/update/`*Wartungspaket*`.install`, in dem sich die Installationsprotokolldateien befinden.

WebSphere eXtreme Scale deinstallieren

Wenn Sie WebSphere eXtreme Scale aus Ihrer Umgebung entfernen möchten, können Sie dazu den Assistenten verwenden, oder Sie können das Produkt im unbeaufsichtigten Modus deinstallieren.

Vorbereitende Schritte

Achtung: Das Deinstallationsprogramm entfernt alle Binärdateien und alle Wartungspakete, wie z. B. Fixpacks und vorläufige Fixes, gleichzeitig.

Vorgehensweise

1. Stoppen Sie alle Prozesse, in denen eXtreme Scale ausgeführt wird.

   Vorsicht:
   Stellen Sie sicher, dass alle aktiven Prozesse gestoppt sind. Wenn die aktiven Prozesse nicht gestoppt sind, wird die Deinstallation fortgesetzt, was zu unvorhersehbaren Ereignissen führen und die Deinstallation auf einigen Plattformen in einem unbestimmten Zustand hinterlassen kann.

   - Wenn Sie eXtreme Scale im eigenständigen Modus installiert haben, verwenden Sie zum Stoppen von Prozessen die Informationen unter "Eigenständige Server stoppen".
   - Wenn Sie eXtreme Scale mit einer vorhandenen Installation von WebSphere Application Server installiert haben, finden Sie unter "Befehlszeilendienstprogramme" weitere Informationen zum Stoppen von Prozessen von WebSphere Application Server.
2. Deinstallieren Sie das Produkt. Sie können die Deinstallation in einer grafischen Benutzerschnittstelle oder im unbeaufsichtigten Modus ausführen.

   Anmerkung: Wenn Sie die Antwortdatei `wxssetup.response` für unbeaufsichtigte oder GUI-Installationen bzw. -Deinstallationen verwenden, müssen Sie immer den vollständig qualifizierten Pfad angeben. Die Antwortdatei ist bei der GUI-Deinstallation optional.

   - **Gehen Sie zum Ausführen der Deinstallation über die grafische Benutzerschnittstelle (GUI) wie folgt vor:**
     - Linux UNIX *<Ausgangsverzeichnis_der_Installation>*`/uninstall_wxs/uninstall`
     - Windows *<Ausgangsverzeichnis_der_Installation>*`\uninstall_wxs\uninstall.exe`

Wenn Sie die Deinstallation über die grafische Benutzerschnittstelle und die Datei `wxssetup.response` ausführen möchten, verwenden Sie einen der folgenden Befehle:

– Linux UNIX

```
<Ausgangsverzeichnis_der_Installation>/uninstall_wxs/uninstall -options
<erforderlicher_vollständiger_Installationspfad>/wxssetup.response
```

– Windows

```
<Ausgangsverzeichnis_der_Installation>\uninstall_wxs\uninstall.exe -options
<erforderlicher_vollständiger_Installationspfad>\wxssetup.response
```

- **Gehen Sie zum Ausführen der Deinstallation im unbeaufsichtigten Modus unter Verwendung des Antwortdateiscripts `wxssetup.response` wie folgt vor:**

  – Linux UNIX

  ```
  <Ausgangsverzeichnis_der_Installation>/uninstall_wxs/uninstall -options
  <erforderlicher_vollständiger_Installationspfad>/wxssetup.response -silent
  ```

  – Windows

  ```
  <Ausgangsverzeichnis_der_Installation>\uninstall_wxs\uninstall.exe -options
  <erforderlicher_vollständiger_Installationspfad>\wxssetup.response -silent
  ```

Ergebnisse

Sie haben eXtreme Scale aus Ihrer Umgebung entfernt.

Kapitel 4. WebSphere eXtreme Scale for z/OS anpassen

Mit WebSphere Customization Tools können Sie angepasste Jobs generieren und ausführen, um WebSphere eXtreme Scale for z/OS anzupassen.

Vorbereitende Schritte

- Vergewissern Sie sich, dass Ihr System die aktuelle Version von WebSphere Application Server Network Deployment enthält:
  - Wenn Sie mit Version 6.1 arbeiten, muss Ihr System mindestens Fixpack 31 enthalten. Weitere Informationen finden Sie unter "Anwendungsserverumgebung der Version 6.1 installieren".
  - Wenn Sie mit Version 7.0 arbeiten, muss Ihr System mindestens Fixpack 9 enthalten. Weitere Informationen finden Sie unter "Anwendungsserverumgebung der Version 7.0 installieren".
- Installieren Sie WebSphere eXtreme Scale for z/OS. Weitere Informationen finden Sie unter *WebSphere eXtreme Scale Program Directory* auf der Bibliothekswebseite von WebSphere eXtreme Scale.

Informationen zu diesem Vorgang

Generieren Sie mit WebSphere Customization Tools Anpassungsdefinitionen. Laden Sie die angepassten Jobs hoch, und führen Sie sie aus, um WebSphere eXtreme Scale for z/OS anzupassen. Weitere Informationen finden Sie in den folgenden Abschnitten:

Vorgehensweise

- „WebSphere Customization Tools installieren”
- „Anpassungsdefinitionen generieren” auf Seite 63
- „Angepasste Jobs hochladen und ausführen” auf Seite 64

WebSphere Customization Tools installieren

Installieren Sie WebSphere Customization Tools Version 7.0.0.6 oder höher, um Ihre Umgebung von WebSphere eXtreme Scale for z/OS anzupassen.

Vorbereitende Schritte

Installieren Sie WebSphere eXtreme Scale for z/OS. Weitere Informationen finden Sie unter *WebSphere eXtreme Scale Program Directory* auf der Bibliotheksseite.

Informationen zu diesem Vorgang

WebSphere Customization Tools ist ein workstationbasiertes Grafiktool, mit dem Sie angepasste Jobs erstellen, die Laufzeitumgebungen für WebSphere eXtreme Scale for z/OS einrichten.

Vorgehensweise

1. Verwenden Sie FTP, um die Erweiterungsdateien `xs.wct` und `xspf.wct` von Ihrem z/OS-System auf die Workstation zu kopieren, auf der Sie WebSphere Customization Tools installieren. Die Erweiterungsdateien befinden sich im Verzeichnis `/usr/lpp/zWebSphereXS/util/V7R1/WCT` auf Ihrem z/OS-System.

2. Laden Sie WebSphere Customization Tools Version 7.0.0.6 oder höher von der entsprechenden Website herunter, und installieren Sie das Produkt:
   - Windows WebSphere Customization Tools for Windows
   - Linux WebSphere Customization Tools for Linux®
3. Laden Sie die Datei `xs.wct` in die Anwendung WebSphere Customization Tools hoch.
   a. Starten Sie die Anwendung WebSphere Customization Tools auf Ihrer Workstation.
   b. Klicken Sie auf **Help** → **Software Updates** → **Install Extension**.
   c. Klicken Sie in der Anzeige "WebSphere Customization Tools Extension Locations" auf **Install new extension location**.
   d. Klicken Sie in der Anzeige "Source Archive File" auf **Browse**, navigieren Sie zu dem Verzeichnis, in das Sie die Datei `xs.wct` in Schritt 1 kopiert haben, und klicken Sie auf **Open**.
   e. Klicken Sie in der Anzeige "Summary" auf **Next**.

      Anmerkung: Die Anzeige "Install Successful" erscheint. Bevor Sie auf **Finish** klicken können, müssen Sie die Daten aus dem Feld "Location" kopieren und speichern:

      ```
      C:\Documents and Settings\Administrator\WCT\workspace\configuration\
      com.ibm.ws.pmt.update\com.ibm.ws390.pmt.xs_7.1.0.0\eclipse
      ```

   f. Klicken Sie in der Anzeige "Product Configuration" auf **Add an extension location**. Fügen Sie die Daten, die Sie im vorherigen Schritt kopiert haben, in das Feld "Location" ein, und klicken Sie anschließend auf **OK**.
   g. Klicken Sie auf **Yes**, um WebSphere Customization Tools erneut zu starten.
4. Laden Sie die Datei `xspf.wct` in die Anwendung WebSphere Customization Tools hoch.
   a. Klicken Sie auf **Help** → **Software Updates** → **Install Extension**.
   b. Klicken Sie in der Anzeige "WebSphere Customization Tools Extension Locations" auf **Install new extension location**.
   c. Klicken Sie in der Anzeige "Source Archive File" auf **Browse**, navigieren Sie zu dem Verzeichnis, in das Sie die Datei `xspf.wct` in Schritt 1 kopiert haben, und klicken Sie auf **Open**.
   d. Klicken Sie in der Anzeige "Summary" auf **Next**.

      Anmerkung: Die Anzeige "Install Successful" erscheint. Bevor Sie auf **Finish** klicken können, müssen Sie die Daten aus dem Feld "Location" kopieren und speichern:

      ```
      C:\Documents and Settings\Administrator\WCT\workspace\configuration\
      com.ibm.ws.pmt.update\com.ibm.ws390.pmt.xs_7.1.0.0\eclipse
      ```

   e. Klicken Sie in der Anzeige "Product Configuration" auf **Add an extension location**. Fügen Sie die Daten, die Sie im vorherigen Schritt kopiert haben, in das Feld "Location" ein, und klicken Sie anschließend auf **OK**.
   f. Klicken Sie auf **Yes**, um WebSphere Customization Tools erneut zu starten.

Nächste Schritte

Nachdem Sie beide Erweiterungsdateien hochgeladen und WebSphere Customization Tools erneut gestartet haben, können Sie mit Profile Management Tool Anpassungsdefinitionen für eXtreme Scale for z/OS generieren. Weitere Informationen finden Sie im Abschnitt „Anpassungsdefinitionen generieren” auf Seite 63.

Anpassungsdefinitionen generieren

Verwenden Sie die Funktion Profile Management Tool in WebSphere Customization Tools, um Anpassungsdefinitionen zu generieren und angepasste Jobs für WebSphere eXtreme Scale for z/OS zu erstellen.

Vorbereitende Schritte

Installieren Sie WebSphere Customization Tools, und laden Sie die Erweiterungsdateien `xs.wct` und `xspf.wct` hoch. Weitere Informationen finden Sie im Abschnitt „WebSphere Customization Tools installieren" auf Seite 61.

Informationen zu diesem Vorgang

Sie können Anpassungsdefinitionen mit Profile Management Tool generieren, das in WebSphere Customization Tools bereitgestellt wird. Eine *Anpassungsdefinition* ist eine Gruppe von Dateien, die zum Erstellen angepasster Jobs für die Konfiguration von WebSphere eXtreme Scale for z/OS verwendet werden.

Vorgehensweise

1. Starten Sie Profile Management Tool.
   - Windows Klicken Sie auf **Start** → **Programme** → **IBM WebSphere** → **WebSphere Customization Tools**. Klicken Sie nach dem Starten der Anwendung auf das Register **Profile Management Tool**.
   - Linux Klicken Sie auf ***Betriebssystemmenüs*** → **IBM WebSphere** → **WebSphere Customization Tools**. Klicken Sie nach dem Starten der Anwendung auf das Register **Profile Management Tool**.
2. Fügen Sie eine vorhandene Position hinzu, oder erstellen Sie eine Position für die Anpassungsdefinition, die Sie erstellen möchten. Klicken Sie auf der Registerkarte **Customization Locations** auf **Add**. Wenn Sie eine Position erstellen, verweist das Feld "Version" auf die vorhandene Version des Produkts WebSphere Application Server, die auf Ihrem z/OS-System installiert ist.

   Anmerkung: Verwenden Sie nicht dieselbe Position, die Sie für andere Anpassungsdefinitionen von eXtreme Scale verwenden.
3. Generieren Sie die Anpassungsdefinition. Klicken Sie auf der Registerkarte **Customization Definitions** auf **Augment**.
4. Wählen Sie den Typ der zu erstellenden Definitionsumgebung aus:
   - Eigenständiger Anwendungsserverknoten
   - Deployment Manager
   - Anwendungsserver
   - Verwalteter (angepasster) Knoten
5. Füllen Sie die Felder in den Anzeigen aus. Geben Sie die Werte für die Parameter an, die zum Erstellen des z/OS-Systems verwendet werden.
6. Klicken Sie auf **Augment**, um die Anpassungsdefinition zu generieren.

Nächste Schritte

Laden Sie den angepassten Job auf Ihr z/OS-Zielsystem hoch. Weitere Informationen finden Sie im Abschnitt „Angepasste Jobs hochladen und ausführen" auf Seite 64.

Angepasste Jobs hochladen und ausführen

Nachdem Sie die Anpassungsdefinitionen generiert haben, können Sie die angepassten Jobs, die den Definitionen zugeordnet sind, auf Ihr System mit WebSphere eXtreme Scale for z/OS hochladen und ausführen.

Vorbereitende Schritte

Generieren Sie die Anpassungsdefinitionen für die Jobs, die Sie auf Ihr z/OS-System hochladen möchten. Weitere Informationen finden Sie im Abschnitt „Anpassungsdefinitionen generieren” auf Seite 63.

Informationen zu diesem Vorgang

Laden Sie die angepassten Jobs, die Sie mit WebSphere Customization Tools für die Verwaltung und Überwachung Ihrer Umgebung von WebSphere eXtreme Scale for z/OS erstellt haben, hoch, und führen Sie sie aus.

Vorgehensweise

1. Laden Sie die angepassten Jobs hoch. Wählen Sie auf der Registerkarte **Customization Definitions** die Jobs aus, die Sie hochladen möchten, und klicken Sie anschließend auf **Process**.
2. Laden Sie die Jobs auf den FTP-Server auf Ihrem z/OS-System hoch. Geben Sie die erforderlichen Informationen in der Anzeige **Upload Customiziation Definition** an.
3. Klicken Sie auf **Finish**.
4. Führen Sie die angepassten Jobs aus. Klicken Sie auf das Register **Customization Instructions**, und folgen Sie den Anpassungsanweisungen für jeden Job.

Kapitel 5. Anwendungsimplementierung planen

Bevor Sie Anwendungen in WebSphere eXtreme Scale implementieren, müssen Sie die Hardware- und Softwarevoraussetzungen, die Netz- und Optimierungseinstellungen, die Implementierungskonfigurationen usw. überprüfen. Sie können auch die Prüfliste für Betriebsbereitschaft verwenden, um sicherzustellen, dass Ihre Umgebung für die Implementierung von Anwendungen bereit ist.

Eine Beschreibung der bewährten Verfahren für das Entwerfen von eXtreme-Scale-Anwendungen finden Sie im folgenden Artikel auf developerWorks: Principles and best practices for building high performing and highly resilient WebSphere eXtreme Scale applications.

Übersicht über die Anwendungsimplementierung

Vor der Verwendung von WebSphere eXtreme Scale in einer Produktionsumgebung sollten Sie sich die folgenden Punkte zur Optimierung der Implementierung ansehen.

Anwendungsimplementierung planen

Die folgende Liste enthält die zu beachtenden Punkte:

- Anzahl der Systeme und Prozessoren: Wie viele physische Maschinen und Prozessoren sind in der Umgebung erforderlich?
- Anzahl der Server: Wie viele eXtreme-Scale-Server sind für die Speicherung der eXtreme-Scale-Maps erforderlich?
- Anzahl der Partitionen: Das in den Maps gespeicherte Datenvolumen ist ein Faktor für die Bestimmung der Anzahl erforderlicher Partitionen.
- Anzahl der Replikate: Wie viele Replikate sind für jedes primäre Shard in der Domäne erforderlich?
- Synchrone oder asynchrone Replikation: Sind die Daten elementar, so dass eine synchrone Replikation erforderlich ist? Oder hat die Leistung eine höhere Priorität, so dass die asynchrone Replikation die richtige Wahl ist?
- Größe der Heap-Speicher: Welches Datenvolumen wird auf jedem Server gespeichert?

Hardware- und Softwarevoraussetzungen

Dieser Abschnitt enthält eine Übersicht über die Hardware- und Betriebssystemvoraussetzungen. Obwohl Sie keine bestimmte Version der Hardware oder des Betriebssystems für WebSphere eXtreme Scale verwenden müssen, können Sie sich die detaillierte Liste der für jedes Betriebssystem offiziell unterstützten Hardware- und Softwareoptionen auf der Seite "System Requirements" der Produktunterstützungswebsite ansehen. Sollten die im Information Center enthaltenen Informationen und die Informationen auf der Webseite zu den Systemvoraussetzungen widersprüchlich sein, haben die Informationen auf der Website Vorrang. Die Informationen zu den Voraussetzungen im Information Center werden nur im Hinblick auf die Bedienerfreundlichkeit bereitgestellt.

Rufen Sie die Webseite "System Requirements" auf.

Hardwarevoraussetzungen

WebSphere eXtreme Scale setzt keine bestimmte Hardwareversion voraus. Die Hardwarevoraussetzungen richten sich nach der unterstützten Hardware für die Installation der Java Platform, Standard Edition, die Sie für die Ausführung von WebSphere eXtreme Scale verwenden. Wenn Sie eXtreme Scale mit WebSphere Application Server oder einer anderen Java-EE-Implementierung (Java Platform, Enterprise Edition) verwenden, sind die Hardwarevoraussetzungen dieser Plattformen für WebSphere eXtreme Scale ausreichend.

Betriebssystemvoraussetzungen

eXtreme Scale setzt keine bestimmte Betriebssystemversion voraus. Jede Java-SE- und jede Java-EE-Implementierung setzt verschiedene Betriebssystemversionen oder -Fixes für Probleme voraus, die während des Testens der Java-Implementierung erkannt werden. Die von diesen Implementierungen vorausgesetzten Versionen sind für eXtreme Scale ausreichend.

Voraussetzungen in Bezug auf WebSphere Application Server

Clients und Server von eXtreme Scale, die in einer verteilten Umgebung ausgeführt werden, und lokale speicherinterne ObjectGrids werden in WebSphere Application Server Version 6.0.2 und höher unterstützt.

Anmerkung: Wenn Sie den dynamischen Cacheprovider verwenden möchten, muss eine der folgenden Mindestvoraussetzungen erfüllt sein:

- WebSphere Application Server Version 6.1.0.25 und höher mit vorläufigem Fix PK85622
- WebSphere Application Server Version 7.0.0.3 und höher mit vorläufigem Fix PK85622

Weitere Informationen finden Sie auf der Webseite mit den empfohlenen Fixes für WebSphere Application Server.

Weitere Voraussetzungen für den Anwendungsserver

Andere Java-EE-Implementierungen können die Laufzeitumgebung von eXtreme Scale als lokale Instanz oder als Client für Server von eXtreme Scale verwenden. Für die Implementierung von Java SE müssen Sie Version 1.4.2 oder höher verwenden.

Hinweise zu Java Platform, Enterprise Edition

Bei der Vorbereitung der Integration von WebSphere eXtreme Scale in eine Java-EE-Umgebung müssen Sie bestimmte Punkte berücksichtigen, wie z. B. Konfigurationsoptionen, Voraussetzungen und Einschränkungen sowie Anwendungsimplementierung und -management.

eXtreme-Scale-Anwendungen in einer Java-EE-Umgebung ausführen

Eine Java-EE-Anwendung kann eine Verbindung zu einer fernen eXtreme-Scale-Anwendung herstellen. Außerdem unterstützt die Umgebung von WebSphere Application Server das Starten eines eXtreme-Scale-Servers beim Starten einer Anwendung im Anwendungsserver.

Wenn Sie eine XML-Datei zum Erstellen einer ObjectGrid-Instanz verwenden und die XML-Datei im Modul der EAR-Datei enthalten ist, greifen Sie mit der Methode getClass().getClassLoader().getResource("META-INF/objGrid.xml") auf die Datei zu, um ein URL-Objekt für die Erstellung einer ObjectGrid-Instanz abzurufen. Setzen Sie im Methodenaufruf den Namen der XML-Datei ein, die Sie verwenden.

Sie können Startup-Beans für eine Anwendung verwenden, um eine ObjectGrid-Instanz beim Starten einer Anwendung zu booten und um die Instanz beim Stoppen der Anwendung zu löschen. Eine Startup-Bean ist eine Stateless Session-Bean mit einer fernen com.ibm.websphere.startupservice.AppStartUpHome-Position und einer fernen com.ibm.websphere.startupservice.AppStartUp-Schnittstelle. Die ferne Schnittstelle hat zwei Methoden: die Methode "start" und die Methode "stop". Verwenden Sie die Methode "start", um die Instanz zu booten, und die Methode "stop", um die Instanz zu löschen. Die Anwendung verwendet die Methode "ObjectGridManager.getObjectGrid", um eine Referenz auf die Instanz zu verwalten. Weitere Informationen finden Sie in den Informationen zum Zugriff auf ein ObjectGrid mit ObjectGridManager im *Programmierhandbuch*.

Klassenladeprogramme verwenden

Wenn Anwendungsmodule, die unterschiedliche Klassenladeprogramme verwenden, eine einzige ObjectGrid-Instanz in einer Java-EE-Anwendung gemeinsam nutzen, müssen Sie sicherstellen, dass die Objekte, die in eXtreme Scale gespeichert werden, und die Plug-ins für das Produkt in einem gemeinsamen Loader der Anwendung enthalten sind.

Lebenszyklus von ObjectGrid-Instanzen in einem Servlet verwalten

Für die Verwaltung des Lebenszyklus einer ObjectGrid-Instanz in einem Servlet können Sie die Methode "init" verwenden, um die Instanz zu erstellen, und die Methode "destroy", um die Instanz zu entfernen. Wenn die Instanz zwischengespeichert ist, wird sie im Servlet-Code abgerufen und bearbeitet. Weitere Einzelheiten finden Sie in den Informationen zum Zugriff auf ein ObjectGrid mit ObjectGridManager im *Programmierhandbuch*.

Caching-Topologie: Speicherinternes und verteiltes Caching

Mit WebSphere eXtreme Scale kann Ihre Architektur speicherinternes Daten-Caching oder verteiltes Client/Server-Daten-Caching verwenden.

WebSphere eXtreme Scale erfordert für den Betrieb eine minimale zusätzliche Infrastruktur. Die Infrastruktur setzt sich aus Scripts für die Installation, das Starten und Stoppen von Java-EE-Anwendungen in einem Server zusammen. Die zwischengespeicherten Daten werden im Server von eXtreme Scale gespeichert, und Clients stellen über Fernzugriff eine Verbindung zum Server her.

Verteilte Caches bieten eine bessere Leistung, Verfügbarkeit und Skalierbarkeit und können unter Verwendung dynamischer Topologien konfiguriert werden, in denen Server automatisch verteilt werden. Zusätzliche Server können hinzugefügt werden, ohne die vorhandenen eXtreme Scale-Server erneut starten zu müssen. Sie können einfache Implementierungen erstellen oder große Implementierungen in Terabytegröße, in denen Tausende von Servern erforderlich sind.

Lokaler Speichercache

Im einfachsten Fall kann eXtreme Scale als lokaler (nicht verteilter) speicherinterner Daten-Grid-Cache verwendet werden. Dies kann insbesondere für Anwendungen mit sehr vielen gemeinsamen Zugriffen von Vorteil sein, in denen mehrere Threads auf transiente Daten zugreifen und diese ändern müssen. Die in einem lokalen eXtreme-Scale-Grid gespeicherten Daten können indexiert und über die Abfrageunterstützung von WebSphere eXtreme Scale abgerufen werden. Die Möglichkeit, die Daten abzufragen, kann Entwicklern bei der Arbeit mit großen speicherinternen Datenmengen besser helfen, als es die eingeschränkte Unterstützung für Datenstrukturen der Java Virtual Machine (JVM) tut, die sofort einsatzfähig ist.

Die lokale speicherinterne Cachetopologie für eXtreme Scale wird verwendet, um einen konsistenten, transaktionsorientierten Zugriff auf temporäre Daten in einer einzelnen Java Virtual Machine zu unterstützen.

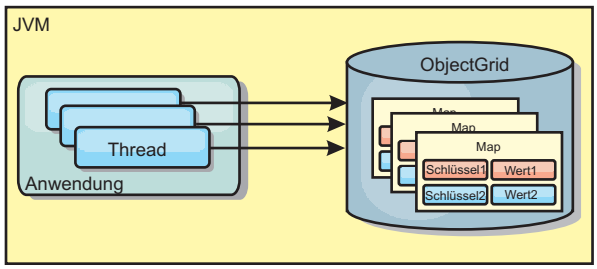


Abbildung 1. Szenario mit einem lokalen speicherinternen Speichercache

Vorteile

- Einfaches Setup: Ein ObjectGrid kann programmgesteuert oder deklarativ über die ObjectGrid-XML-Implementierungsdeskriptordatei oder mit anderen Frameworks wie Spring erstellt werden.
- Schnell: Jede BackingMap kann gesondert für eine optimale Speicherauslastung und gemeinsamen Zugriff optimiert werden.
- Ideal für Topologien mit einer einzigen JVM und einer kleinen Dateigruppe oder für das Caching von Daten, auf die häufig zugegriffen wird.
- Transaktionsorientiert: BackingMap-Aktualisierungen können zu einer einzigen Arbeitseinheit gruppiert und als letzter Teilnehmer in zweiphasige Transaktionen wie JTA-Transaktionen (Java Transaction Architecture) integriert werden.

Nachteile

- Keine Fehlertoleranz.
- Die Daten werden nicht repliziert. Speichercaches eignen sich am besten für schreibgeschützte Referenzdaten.
- Keine Skalierbarkeit. Die für die Datenbank erforderliche Speicherkapazität kann die JVM möglicherweise nicht bereitstellen.
- Es treten Probleme auf, wenn JVMs hinzugefügt werden.
  - Die Daten sind nicht so einfach partitionierbar.
  - Der Status muss in den JVMs manuell repliziert werden, da die einzelnen Cacheinstanzen ansonsten verschiedene Versionen derselben Daten enthalten könnten.
  - Das Ungültigmachen von Einträgen ist kostenintensiv.

– Jeder Cache muss einzeln vorbereitet werden. Die Vorbereitungs- oder Aufwärmphase ist der Zeitraum, in dem eine Gruppe von Daten geladen wird, damit der Cache mit gültigen Daten gefüllt wird.

Einsatz

Die Implementierungstopologie mit dem lokalen Speichercache sollte nur verwendet werden, wenn die Menge der zwischenzuspeichernden Daten klein ist (in eine einzige JVM passt) und relativ stabil ist. Bei diesem Ansatz müssen veraltete Daten toleriert werden. Die Verwendung von Evictor (Bereinigungsprogramm), um nur die am häufigsten verwendeten oder die zuletzt verwendeten Daten im Cache zu verwalten, kann dabei helfen, den Cache klein zu halten und die Relevanz der Daten zu erhöhen.

Auf Peers replizierter lokaler Speichercache

Bei einem lokalen eXtreme-Scale-Cache müssen Sie sicherstellen, dass der Cache synchronisiert wird, wenn mehrere Prozesse mit voneinander unabhängigen Cacheinstanzen vorhanden sind. Hierfür aktivieren Sie einen replizierten Peer-Cache mit JMS.

WebSphere eXtreme Scale enthält zwei Plug-ins, die Transaktionsänderungen automatisch zwischen Peer-ObjectGrid-Instanzen weitergeben. Das JMSObjectGridEventListener-Plug-in gibt eXtreme-Scale-Änderungen automatisch über Java Messaging Service (JMS) weiter.

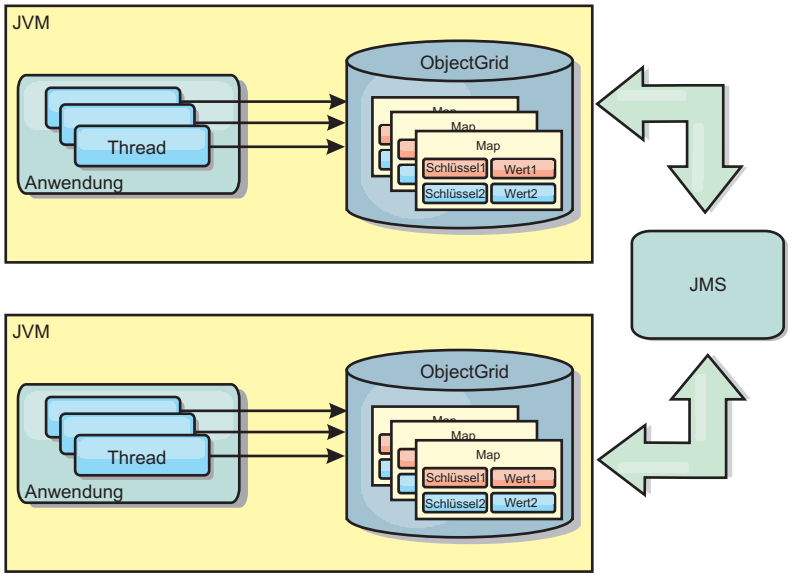


Abbildung 2. Auf Peers replizierter Cache mit Änderungen, die über JMS weitergegeben werden

Wenn Sie eine Umgebung von WebSphere Application Server ausführen, ist auch das TranPropListener-Plug-in verfügbar. Das TranPropListener-Plug-in verwendet den High Availability Manager (kurz HA-Manager), um die Änderungen an jede Peer-Cacheinstanz von eXtreme Scale weiterzugeben.

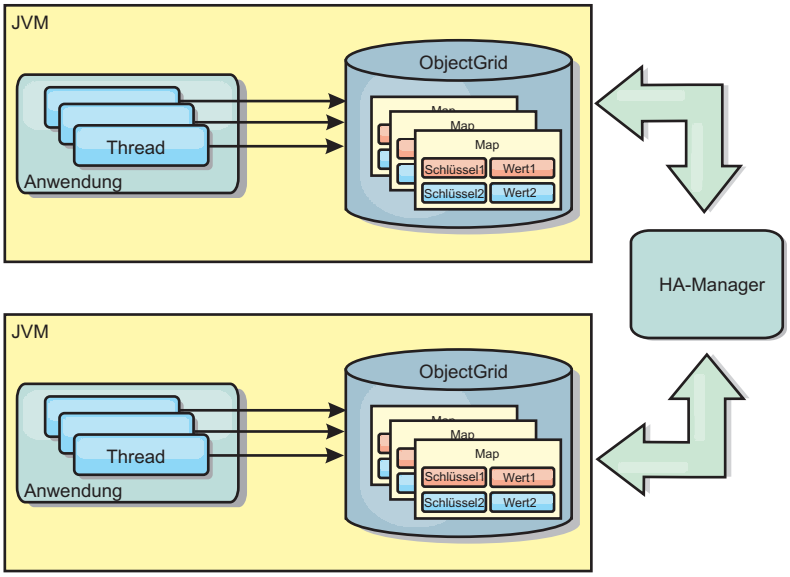


Abbildung 3. Auf Peers replizierter Cache mit Änderungen, die über den High Availability Manager weitergegeben werden

Vorteile

- Die Gültigkeit der Daten ist höher, weil sie häufiger aktualisiert werden.
- Mit dem TranPropListener-Plug-in kann eXtreme Scale wie die lokale Umgebung über das Programm oder deklarativ über die XML-Implementierungsdeskriptordatei von eXtreme Scale oder mit anderen Frameworks wie Spring erstellt werden. Die Integration mit dem High Availability Manager erfolgt automatisch.
- Jede BackingMap kann gesondert für eine optimale Speicherauslastung und gemeinsamen Zugriff optimiert werden.
- BackingMap-Aktualisierungen können zu einer einzigen Arbeitseinheit gruppiert und als letzter Teilnehmer in zweiphasige Transaktionen wie JTA-Transaktionen (Java Transaction Architecture) integriert werden.
- Ideal für Topologien mit wenigen JVMs und einer angemessen kleinen Dateigruppe oder für das Caching von Daten, auf die häufig zugegriffen wird.
- Änderungen an eXtreme Scale werden in allen Peer-Instanzen von eXtreme Scale repliziert. Die Änderungen sind so lange konsistent, wie eine permanente Subskription verwendet wird.

Nachteile

- Die Konfiguration und Verwaltung für JMSObjectGridEventListener kann komplex sein. eXtreme Scale kann über das Programm oder deklarativ über die XML-Implementierungsdeskriptordatei von eXtreme Scale oder mit anderen Frameworks wie Spring erstellt werden.
- Nicht skalierbar: Die für die Datenbank erforderliche Speicherkapazität kann die JVM möglicherweise nicht bereitstellen.
- Funktioniert nicht ordnungsgemäß, wenn Java Virtual Machines hinzugefügt werden:
  - Die Daten sind nicht so einfach partitionierbar.
  - Das Ungültigmachen von Einträgen ist kostenintensiv.
  - Jeder Cache muss einzeln vorbereitet werden.

Einsatz

Diese Implementierungstopologie sollte nur verwendet werden, wenn das zwischenzuspeichernde Datenvolumen gering (d. h. in eine einzige JVM passt) und relativ stabil ist.

Verteilter Cache

In den meisten Fällen wird WebSphere eXtreme Scale als gemeinsam genutzter Cache verwendet, um einen transaktionsgesteuerten Zugriff auf Dateien durch mehrere Komponenten zu ermöglichen, wo ansonsten eine traditionelle Datenbank verwendet werden würde. Bei der Verwendung eines gemeinsam genutzten Caches muss keine Datenbank konfiguriert werden.

Der Cache ist kohärent, weil alle Clients dieselben Daten im Cache sehen. Jede einzelne Information wird im Cache eines einzigen Servers gespeichert. Auf diese Weise werden unnötige Datensatzkopien verhindert, die potenziell verschiedene Versionen der Daten enthalten könnten. Ein kohärenter Cache kann außerdem mehr Daten aufnehmen, wenn dem Grid weitere Server hinzugefügt werden. Die Skalierung des Caches erfolgt linear zum Wachstum des Grids. Da Clients über Prozedurfernaufrufe auf Daten in diesem Grid zugreifen, wird der Cache auch als ferner Cache bezeichnet. Durch die Datenpartitionierung enthält jeder Prozess einen eindeutigen Teil der Gesamtdatengruppe. Größere Grids können mehr Daten aufnehmen und und mehr Anforderungen für diese Daten bearbeiten. Aufgrund der Kohärenz müssen auch keine Daten zum Ungültigmachen im Grid verteilt werden, weil es keine veralteten Daten gibt. Der kohärente Cache enthält jeweils nur die aktuelle Kopie jeder Information.

Wenn Sie in einer Umgebung mit WebSphere Application Server arbeiten, ist auch das TranPropListener-Plug-in verfügbar. Das TranPropListener-Plug-in verwendet die Komponente "High Availability Manager" (kurz HA-Manager) von WebSphere Application Server, um die Änderungen an die einzelnen Peer-ObjectGrid-Cacheinstanzen weiterzugeben.

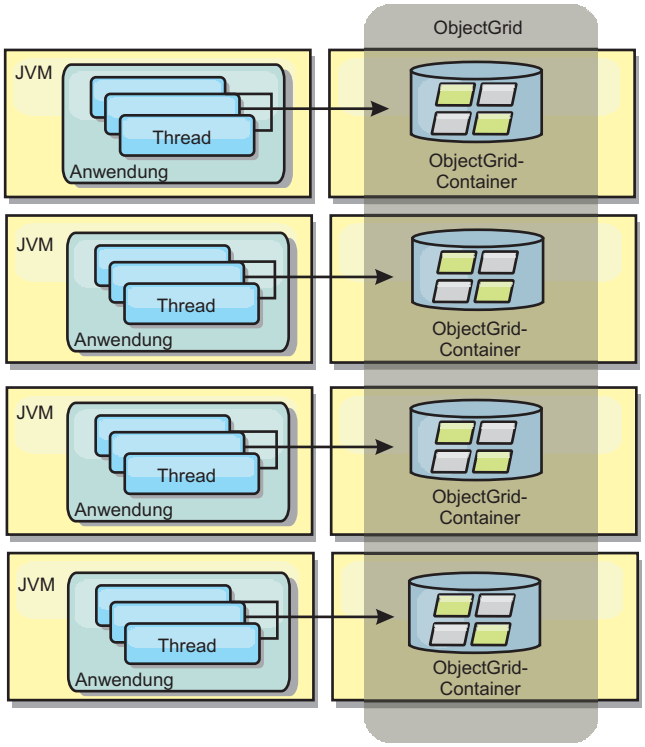


Abbildung 4. Verteilter Cache

Naher Cache

Clients können optional einen lokalen integrierten Cache haben, wenn eXtreme Scale in einer verteilten Topologie verwendet wird. Dieser optionale Cache wird als naher Cache bezeichnet. Er ist ein unabhängiges ObjectGrid in jedem Client, das als Cache für den fernen serverseitigen Cache dient. Der nahe Cache wird standardmäßig aktiviert, wenn eine optimistische Sperrstrategie oder keine Sperrstrategie konfiguriert ist, und kann nicht verwendet werden, wenn eine pessimistische Sperrstrategie konfiguriert ist.

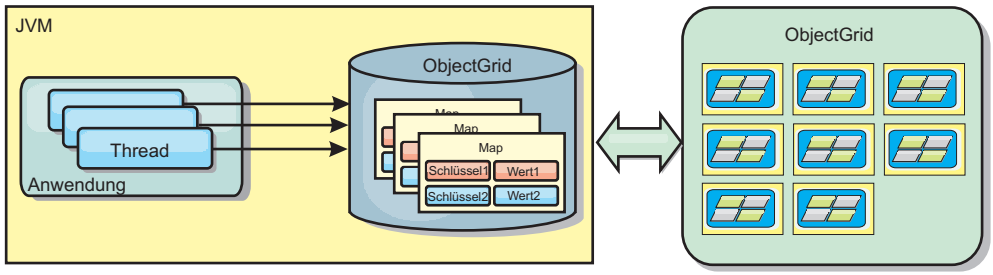


Abbildung 5. Naher Cache

Ein naher Cache ist sehr schnell, weil er den speicherinternen Zugriff auf einen Teil der gesamten zwischengespeicherten Daten ermöglicht, die fern in den Servern von eXtreme Scale gespeichert sind. Der nahe Cache ist nicht partitioniert und enthält Daten aus allen fernen eXtreme-Scale-Partitionen. WebSphere eXtreme Scale kann bis zu drei Cacheschichten haben. Diese sind im Folgenden erläutert:

1. Der Cache auf der Transaktionsschicht enthält alle Änderungen für eine einzelne Transaktion. Der Transaktionscache enthält eine Arbeitskopie der Daten, bis

die Transaktion festgeschrieben wird. Wenn eine Clienttransaktion Daten aus einer ObjectMap anfordert, wird zuerst die Transaktion geprüft.

2. Der nahe Cache auf der Clientschicht enthält einen Teil der Daten aus der Serverschicht. Wenn die Daten nicht auf Transaktionsschicht zu finden sind, werden die Daten (sofern verfügbar) aus dem nahen Cache abgerufen und in den Transaktionscache eingefügt.
3. Das Grid auf der Serverschicht enthält den Hauptteil der Daten und wird von allen Clients gemeinsam genutzt. Die Serverschicht kann partitioniert werden, was die Zwischenspeicherung großer Datenvolumen ermöglicht. Wenn der nahe Cache des Clients die Daten nicht enthält, werden die Daten von der Serverschicht abgerufen und in den Clientcache eingefügt. Die Serverschicht kann auch ein Loader-Plug-in (Ladeprogramm) haben. Wenn das Grid die angeforderten Daten nicht enthält, wird der Loader aufgerufen, der die Daten aus dem Back-End-Datenspeicher abruft und in das Grid einfügt.

Zum Inaktivieren des nahen Caches setzen Sie das Attribut "numberOfBuckets" in der eXtreme-Scale-Deskriptorkonfiguration für das Überschreiben des Clients auf "0". Einzelheiten zu den Sperrstrategien von eXtreme Scale finden Sie im Abschnitt zum Sperren von Map-Einträgen. Der nahe Cache kann auch mit einer gesonderten Bereinigungsrichtlinie und anderen Plug-ins konfiguriert werden, die die eXtreme-Scale-Deskriptorkonfiguration für das Überschreiben des Clients verwenden.

Vorteil

- Schnelle Antwortzeiten, weil alle Zugriffe auf die Daten lokal erfolgen.

Nachteile

- Veraltete Daten bleiben länger verfügbar.
- Es muss ein Evictor (Bereinigungsprogramm) zum Ungültigmachen von Daten verwendet werden, um Speicherengpässe zu verhindern.

Einsatz

Verwenden Sie diese Technik, wenn die Antwortzeiten wichtig und veraltete Daten tolerierbar sind.

Integrierter Cache

eXtreme-Scale-Grids können in vorhandenen Prozessen als integrierte eXtreme-Scale-Server ausgeführt oder als externe Prozesse verwaltet werden. Integrierte Grids sind hilfreich, wenn Sie mit einem Anwendungsserver wie WebSphere Application Server arbeiten. Sie können eXtreme-Scale-Server, die nicht integriert sind, über Befehlszeilenscripts starten und in einem Java-Prozess ausführen.

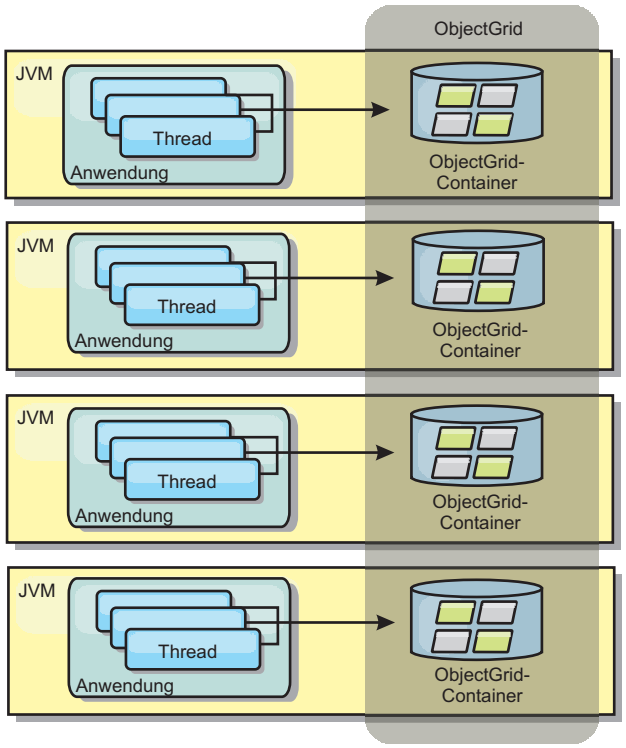


Abbildung 6. Integrierter Cache

Vorteile

- Vereinfachte Verwaltung, da weniger Prozesse zu verwalten sind
- Vereinfachte Anwendungsimplementierung, weil das Grid den Klassen-Loader der Clientanwendung verwendet
- Unterstützung von Partitionierung und hoher Verfügbarkeit

Nachteile

- Erhöhter Speicherbedarf im Clientprozess, weil alle Daten im Prozess erfasst werden
- Erhöhte CPU-Auslastung für die Bearbeitung von Clientanforderungen
- Erschwerte Verarbeitung von Anwendungsupgrades, da Clients dieselben Java-Anwendungsarchivdateien wie die Server verwenden
- Weniger Flexibilität. Die Skalierung von Clients und Grid-Servern ist nicht linear. Wenn Server extern definiert werden, haben Sie mehr Flexibilität bei der Verwaltung der Prozessanzahl.

Einsatz

Verwenden Sie integrierte Grids, wenn im Clientprozess reichlich Speicher für die Grid-Daten und potenzielle Failover-Daten frei ist.

Weitere Informationen finden Sie im Abschnitt zum Mechanismus für Clientinaktivierung im *Administratorhandbuch*.

Multmaster-Replikationstopologie (AP)

Wenn Sie das asynchrone Multimaster-Replikationsfeature verwenden, können zwei oder mehr Grids exakte Spiegel voneinander werden. Diese Spiegelung wird

durch asynchrone Replikation zwischen Links erreicht werden, die die Grids miteinander verbinden. Jedes Grid ist in einer vollständig unabhängigen "Domäne" enthalten, hat einen eigenen Katalogservice, eigene Containerserver und einen eindeutigen Domänennamen. Mit dem asynchronen Multimaster-Replikationsfeature können Sie Links verwenden, um eine Sammlung dieser Domänen miteinander zu verbinden, und diese Domänen anschließend durch Replikation über die Links synchronisieren. eXtreme Scale ermöglicht Ihnen, nahezu jede Topologie zu erstellen, weil die Definition der Links zwischen den Domänen Ihnen überlassen bleibt.

7.1+ Die Multimaster-Grid-Replikation ist ein wichtiges neues Feature in Version 7.1.

Domänen: Grids mit speziellen Merkmalen

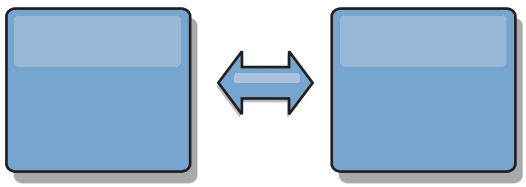
In Multimaster-Replikationstopologien verwendete Grids werden als *Domänen* bezeichnet. Jede Domäne muss die folgenden Merkmale haben:

- Hat einen privaten Katalogservice mit einem eindeutigen Domänennamen
- Hat denselben Grid-Namen wie andere Grids in der Domäne
- Hat dieselbe Anzahl an Partitionen wie andere Grids in der Domäne
- Ist ein Grid vom Typ FIXED_PARTITION (Grids vom Typ PER_CONTAINER können nicht repliziert werden)
- Hat dieselbe Anzahl an Partitionen (kann, muss aber nicht dieselbe Anzahl und dieselben Typen von Replikaten haben)
- Hat dieselben Replikationsdatentypen wie andere Grids in der Domäne
- Hat dieselben MapSet-Namen, Map-Namen und dynamischen Map-Schablonen wie andere Grids in der Domäne

Nachdem die Domänen in der Topologie gestartet wurden, werden alle Grids mit den zuvor beschriebenen Merkmalen repliziert. Beachten Sie, dass die Replikationsrichtlinien der Grids ignoriert werden.

Links, die Domänen verbinden

Eine Grid-Replikationsinfrastruktur ist ein verbundener Graph von Domänen mit bidirektionalen Links zwischen den Domänen. Über einen Link können zwei Domänen Daten miteinander austauschen. Die einfachste Topologie ist beispielsweise ein Domänenpaar mit einem einzigen Link zwischen ihnen. Die Domänen werden von links aus gesehen beginnend mit "A", dann "B" usw. benannt. Der Link kann ein Weitverkehrsnetz (WAN) durchqueren und große Distanzen überwinden. Wenn der Link unterbrochen wird, können trotzdem Änderungen an den Daten in den beiden Domänen vorgenommen werden. Die Änderungen werden später abgeglichen, wenn der Link die Domänen wieder verbindet. Links versuchen nach der Unterbrechung der Netzverbindung automatisch, die Verbindung wiederherzustellen.

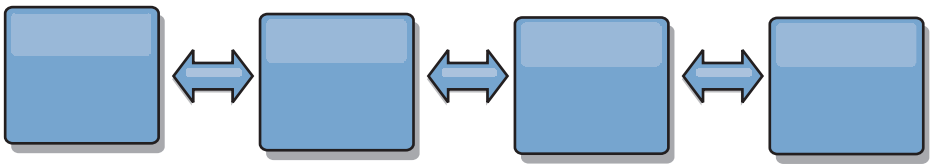


Nach dem Einrichten der Links versucht eXtreme Scale zuerst, alle Domänen abzugleichen, und dann, diese identischen Bedingungen beizubehalten, wenn Änderun-

gen in den Domänen stattfinden. eXtreme Scale hat das Ziel, dass jede Domäne eine exakte Spiegelung jeder anderen Domäne ist, die über die Links verbunden ist. Die Replikationslinks zwischen den Domänen stellen sicher, dass alle Änderungen, die in einer Domäne vorgenommen werden, in die anderen Domänen kopiert werden.

Reihentopologien

Obwohl die Reihentopologie zu den einfachsten Topologien gehört, veranschaulicht sie einige Qualitäten der Links. Zunächst ist es nicht erforderlich, dass eine Domäne direkte mit jeder anderen Domäne verbunden ist, damit sie Änderungen empfängt. Domäne B übernimmt Änderungen von Domäne A. Domäne C empfängt Änderungen von Domäne A über Domäne B, die die Domänen A und B verbindet. Domäne D empfängt Änderungen von den anderen Domänen über Domäne C. Auf diese Weise kann die Quelle der Änderungen von der Verteilung der Änderungen entlastet werden.



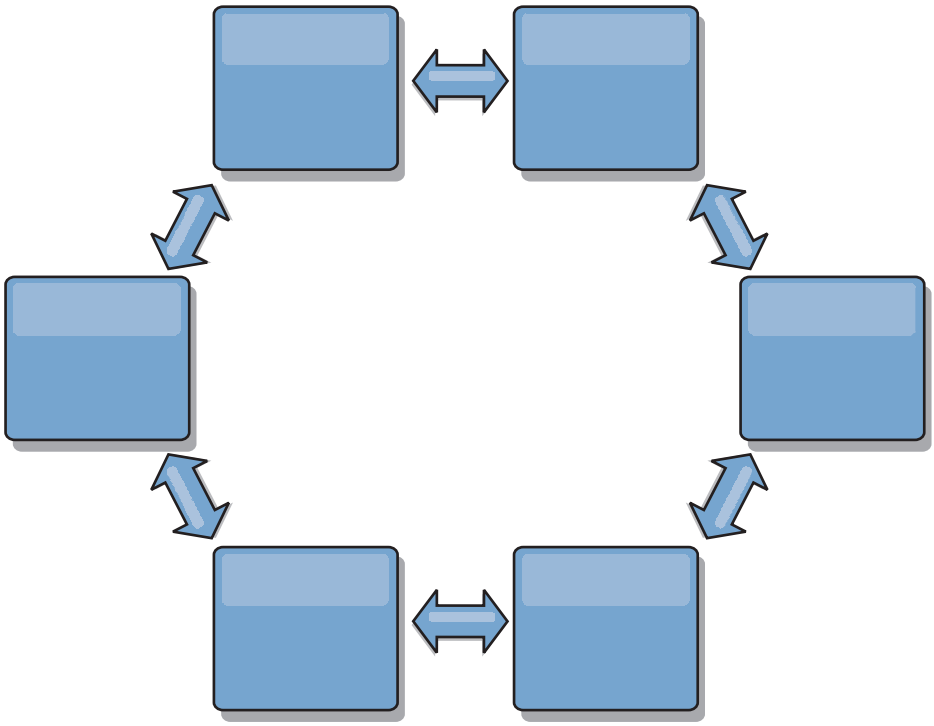
Wenn Domäne C ausfällt, treten die folgenden Ereignisse ein:

1. Domäne D ist verwaist, bis Domäne C erneut gestartet wird.
2. Domäne C synchronisiert sich selbst mit Domäne B, die eine Kopie von Domäne A ist.
3. Domäne D verwendet Domäne C, um sich mit den Änderungen in den Domänen A und B zu synchronisieren, die vorgenommen wurden, während Domäne D verwaist war (aufgrund des Ausfalls von Domäne C).

Am Ende sind die Domänen A, B, C und D wieder identisch.

Ringtopologien

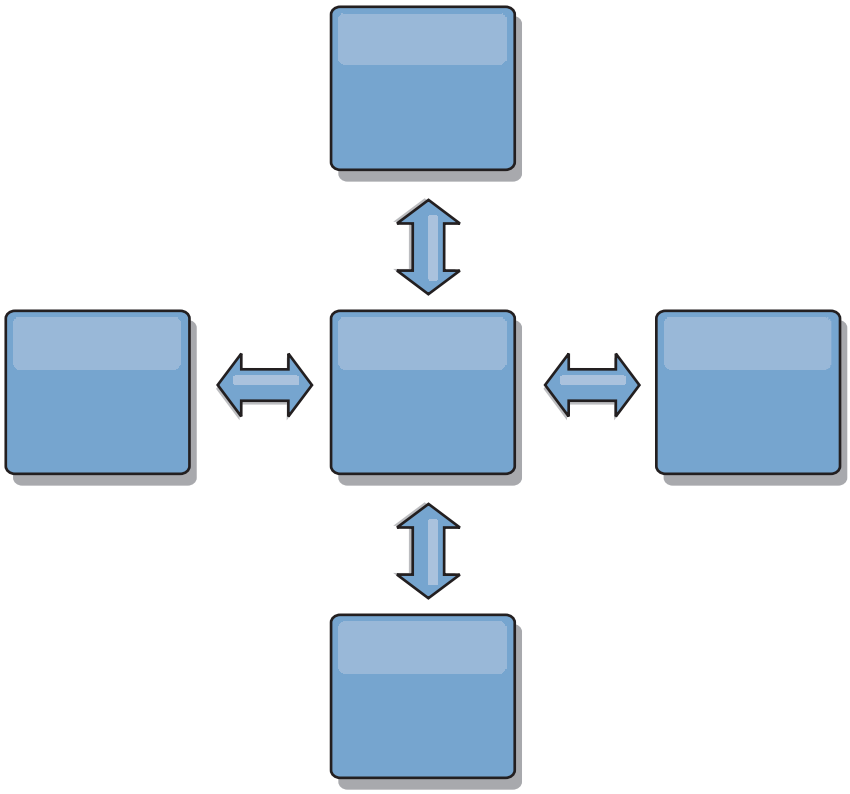
Ringtopologien sind ein Beispiel für eine Topologie mit erhöhter Ausfallsicherheit. Eine Domäne oder ein einzelner Link kann ausfallen, aber die verbleibenden Domänen können trotzdem Änderungen empfangen, weil die Änderungen im Ring in die andere Richtung (von der ausgefallenen Domäne bzw. vom ausgefallenen Link weg) weitergegeben werden. Jede Domäne hat zwei Links zu den anderen Domänen. Jede Domäne hat maximal zwei Links, unabhängig davon, wie groß eine Ringtopologie ist. Die Latenzzeit für die Weitergabe der Änderung kann hoch sein, weil Änderungen einer bestimmten Domäne unter Umständen mehrere Domänen durchqueren müssen, bevor sie von allen Domänen gesehen werden. Eine Reihentopologie hat dasselbe Problem.



Dies ist eine Abbildung einer fortgeschrittenen Ringtopologie mit einer Stammdomäne in der Mitte des Rings. Die Stammdomäne dient als zentrale Clearing-Stelle, während die anderen Domänen als ferne Clearing-Stellen für Änderungen dienen, die in der Stammdomäne vorgenommen werden. Die Stammdomäne kann Änderungen zwischen den Domänen arbitrieren. Wenn eine Ringtopologie mehrere Ringe um die Stammdomäne herum enthält, kann die Stammdomäne Änderungen nur zwischen den Domänen im innersten Ring arbitrieren. Die Ergebnisse der Arbitrierung werden jedoch an die Domänen in den anderen Ringen verteilt.

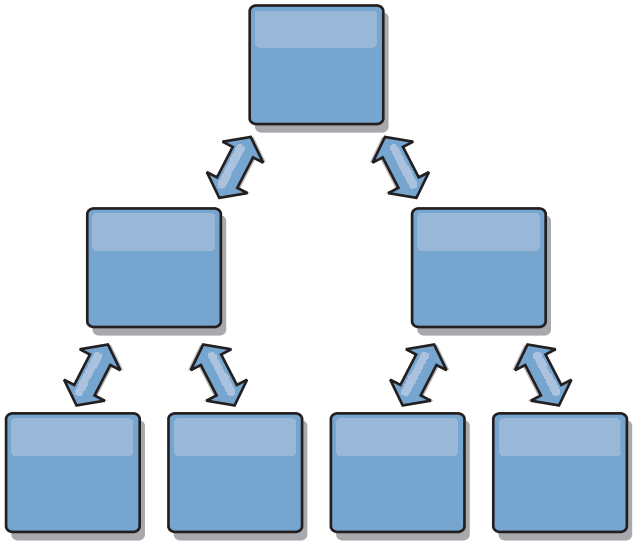
Hub- und Peripherietopologien

Eine Hub- und Peripherietopologie hat bessere Latenzzeiten, d. h., Änderungen durchqueren maximale eine zwischengeordnete Domäne (den Hub), bringt aber andere Probleme mit sich. Eine solche Topologie hat eine zentrale Domäne, die als Hub dient. Diese "Hub-Domäne" ist über einen Link mit jeder "Peripheriedomäne" verbunden. Die Last der Verteilung von Änderungen an die Domänen liegt klar auf dem Hub. Der Hub dient als Clearing-Stelle für Kollisionen. Deshalb kann dieses Setup für bestimmte Szenarien von entscheidender Bedeutung sein. In einer Umgebung mit einer hohen Aktualisierungsrate muss der Hub unter Umständen auf mehr Hardwarekomponenten als die Peripheriedomänen ausgeführt werden, damit er Schritt halten kann. eXtreme Scale ist für eine lineare Skalierung konzipiert, d. h., Sie können den Hub bei Bedarf ohne Schwierigkeit vergrößern. Wenn der Hub jedoch ausfällt, werden die Änderungen erst nach einem Neustart des Hubs wieder verteilt. Alle Änderungen in den Peripheriedomänen werden verteilt, nachdem die Hub-Verbindung wiederhergestellt ist.



Baumtopologien

Ein letztes Topologiebeispiel ist eine azyklische gerichtete Baumstruktur. Azyklisch bedeutet, dass es keine Zyklen und keine Schleifen gibt. Gerichtet bedeutet, dass Links nur zwischen übergeordneten und untergeordneten Elementen existieren. Diese Konfiguration kann für Topologien hilfreich sein, die so viele Domänen haben, dass die Verwendung eines zentralen Hubs, der mit jeder möglichen Peripheriedomäne verbunden ist, nicht praktisch ist, oder in Domänen, in denen Sie untergeordnete Domänen hinzufügen können, ohne die Stammdomäne zu aktualisieren.



Diese Topologie kann trotzdem eine zentrale Clearing-Stelle in der Stammdomäne haben, aber die zweite Ebene kann als ferne Clearing-Stelle für Änderungen die-

nen, die in den Domänen unterhalb dieser Ebene vorgenommen werden. Die Stammdomäne kann Änderungen zwischen den Domänen nur auf der zweiten Ebene arbitrieren. N-gliedrige Baumstrukturen sind ebenfalls möglich. Eine N-gliedrige Baumstruktur hat auf jeder Ebene N untergeordnete Elemente. Jede Domäne hat ein Fanout (Ausgabefächerung) von N.

Arbitrierungshinweise für das Topologiedesign

Änderungskollisionen können auftreten, wenn dieselben Datensätze gleichzeitig an zwei Stellen geändert werden können. Konfigurieren Sie alle Domänen mit denselben Werten für CPU-, Hauptspeicher und Netzressourcen. Sie können beobachten, dass Domänen, die für das Kollisions-Handling (Arbitrierung) zuständig sind, mehr Ressourcen als andere Domänen verbrauchen. Kollisionen werden automatisch erkannt. Sie werden mit einem der folgenden beiden Mechanismen behoben:

- **Standardkollisions-Arbiter**: Standardmäßig werden die Änderungen aus der Domäne verwendet, deren Name in der lexikalischer Reihenfolge am niedrigsten steht. Wenn beispielsweise Domäne A und Domäne B einen Konflikt in Bezug auf einen Datensatz verursachen, wird die Änderung aus Domäne B ignoriert. Domäne A behält ihre Version, und der Datensatz in Domäne B wird geändert, so dass er dem Datensatz aus Domäne A entspricht. Dies gilt auch für Anwendungen, in denen Benutzer oder Sitzungen normalerweise gebunden sind oder eine Affinität zu einem der Grids haben.
- **Angepasster Kollisions-Arbiter:** Anwendungen können einen angepassten Arbiter bereitstellen. Wenn eine Domäne eine Kollision erkennt, ruft sie den Arbiter auf. Informationen zum Entwickeln eines 'guten' angepassten Arbiters finden Sie unter Angepasste Arbiter für Multimaster-Replikation.

Für Topologien, in denen Kollisionen möglich sind, sollten Sie die Verwendung einer Hub- und Peripherietopologie oder einer Baumtopologie in Erwägung ziehen. Diese beiden Topologien sind dienlich, um Endloskollisionen zu vermeiden, die auftreten können, wenn

1. in mehreren Domänen eine Kollision auftritt,
2. jede Domäne die Kollision lokal behebt, was zu Überarbeitungen führt,
3. die Überarbeitungen kollidieren, was zu Überarbeitungen von Überarbeitungen führt,
4. da die Überarbeitungen zwischen den verschiedenen Domänen weitergeleitet werden, um eine Synchronizität zu erreichen.

Zur Vermeidung von Endloskollisionen wählen Sie eine bestimmte Domäne, eine *Arbitrierungsdomäne*, als Kollisions-Handler für einen Teil der Domänen aus. In einer Hub- und Peripherietopologie kann der Hub beispielsweise als Kollisions-Handler verwendet werden. Der Peripheriekollisions-Handler ignoriert alle von den Peripheriedomänen erkannten Kollisionen. Die Hub-Domäne erstellt Überarbeitungen, wodurch Kollisionsüberarbeitungen, die außer Kontrolle geraten, verhindert werden. Die für die Behandlung von Kollisionen zugeordnete Domäne muss einen Link zu allen Domänen haben, für die sie Kollisionen beheben soll. In einer Baumtopologie beheben alle internen übergeordneten Domänen Kollisionen für die ihnen unmittelbar untergeordneten Domänen. Wenn Sie eine Ringtopologie verwendet, ist es nicht möglich, eine einzige Domäne im Ring zu bestimmen, die die Kollisionen beheben soll.

In der folgenden Tabelle sind die kompatiblen Arbitrierungsansätze für die verschiedenen Topologien zusammengefasst.

Tabelle 5. Arbitrierungsansätze. Der folgenden Tabelle können Sie entnehmen, ob Anwendungsarbitrierung mit den verschiedenen Technologien kompatibel ist.

Topologie	Anwendungs-arbitrierung?	Anmerkungen
Eine Reihe von zwei Domänen	Ja	Wählen Sie eine Domäne als Arbiter aus.
Eine Reihe von drei Domänen	Ja	Die mittlere Domäne muss der Arbiter sein. Stellen Sie sich die mittlere Domäne als Hub in einer einfachen Hub- und Peripherietopologie vor.
Eine Reihe von mehr als drei Domänen	Nein	Anwendungsarbitrierung wird nicht unterstützt.
Ein Hub mit N Peripheriedomänen	Ja	Der Hub mit den Links zu allen Peripheriedomänen muss die Arbitrierungsdomäne sein.
Ein Ring mit N Domänen	Nein	Anwendungsarbitrierung wird nicht unterstützt.
Eine azyklische gerichtete Baumstruktur (N-gliedriger Baumstruktur)	Ja	Alle Stammknoten müssen nur die ihnen direkt untergeordneten Knoten arbitrieren.

Linkhinweise für das Topologiedesign

Im Idealfall enthält eine Topologie die Mindestanzahl an Links und optimiert gleichzeitig Kompromisse zwischen Latenzzeit für Änderungen, Fehlertoleranz und Leistungsmerkmale.

- **Latenzzeit bei Änderungen**

  Die Latenzzeit bei Änderungen wird durch die Anzahl zwischengeschalteter Domänen bestimmt, die eine Änderung durchlaufen muss, bevor sie in einer bestimmten Domäne ankommt.

  Eine Topologie weist die beste Latenzzeit bei Änderungen auf, wenn zwischengeschaltete Domänen wegfallen, weil jede Domäne mit jeder anderen Domäne verlinkt wird. Eine Domäne muss jedoch proportional zur Anzahl ihrer Links Replikationsarbeiten ausführen. In großen Topologien kann die reine Anzahl zu definierender Links einen hohen Verwaltungsaufwand darstellen.

  Die Geschwindigkeit, mit der eine Änderung in andere Domänen kopiert wird, richtet sich nach weiteren Faktoren, wie z. B.:

  – CPU und Netzbandbreite in der Quellendomäne,
  – Anzahl zwischengeschalteter Domänen und Links zwischen der Quellen- und der Zieldomäne,
  – CPU- und Netzressourcen, die der Quellendomäne, der Zieldomäne und den zwischengeschalteten Domänen zur Verfügung stehen.

- **Fehlertoleranz**

  Die Fehlertoleranz wird durch die Anzahl der existierenden Pfade zwischen zwei Domänen für die Änderungsreplikation bestimmt.

  Wenn nur ein einziger Link zwischen Domänen existiert und dieser Link ausfällt, werden Änderungen nicht weitergegeben. Wenn der einzige Link von einer Domäne zu einer anderen Domäne über zwischengeschaltete Domänen verläuft, werden die Änderungen nicht weitergegeben, wenn eine der zwischengeschalteten Domänen inaktiv ist.

  Stellen Sie sich eine Reihentopologie mit drei Domänen, A, B und C, vor:

```
A <-> B <-> C
```

Wenn eine der folgenden Bedingungen zutrifft, sieht die Domäne C Änderungen von Domäne A nicht:

- Domäne A ist aktiv, und Domäne B ist inaktiv.
- Der Links zwischen A und B ist inaktiv.
- Der Link zwischen B und C ist inaktiv.

Im Gegensatz dazu kann in einer Ringtopologie jede Domäne Änderungen aus beiden Richtungen übernehmen.

```
A <-> B <-> C <-> zurück zu A
```

Wenn beispielsweise Domäne B inaktiv ist, kann Domäne C trotzdem Änderungen direkt von Domäne A übernehmen.

Ein Hub- und Peripheriedesign ist anfällig, wenn der Hub ausfällt, weil alle Änderungen über den Hub verteilt werden. Es ist jedoch zu bedenken, dass eine einzige Domäne trotzdem ein vollständig fehlertolerantes Grid ist, in dem Fehler wie WAN-Probleme oder Probleme im physischen Rechenzentrum selten auftreten.

- **Leistung**

  Die Anzahl der in einer Domäne definierten Links wirkt sich auf die Leistung aus. Je mehr Links definiert werden, desto mehr Ressourcen werden benötigt, und deshalb kann die Replikationsleistung abnehmen. Die Möglichkeit, Änderungen für Domäne A über andere Domänen zu beziehen, entlastet die Domäne A effektiv von der Replikation ihrer Transaktionen in allen Domänen. *Die Last für die Verteilung der Änderungen in einer Domäne ist auf die Anzahl der von dieser Domäne verwendeten Links beschränkt. Sie hat nicht mit der Anzahl der Domänen in der Topologie zu tun.* Diese Eigenschaft bietet Skalierbarkeit und die Möglichkeit, die Last für die Verteilung von Änderungen auf die Domänen in der Topologie zu verteilen, anstatt diese auf eine einzige Domäne zu konzentrieren.

  Eine Domäne kann Änderungen indirekt über andere Domänen beziehen. Stellen Sie sich eine Reihentopologie mit fünf Domänen vor.

  ```
  A <=> B <=> C <=> D <=> E
  ```

  - A bezieht Änderungen von B, C, D und E über B.
  - B bezieht Änderungen von A und C direkt und Änderungen von D und E über C.
  - C bezieht Änderungen von B und D direkt und Änderungen von A über B und Änderungen von E über D.
  - D bezieht Änderungen von C und E direkt und Änderungen von A und B über C.
  - E bezieht Änderungen von D direkt und Änderungen von A, B und C über D.

  Die Verteilungslast ist in den Domänen A und E am geringsten, weil sie jeweils nur einen Link zu einer einzigen Domäne haben. Die Verteilungslast in den Domänen B, C und D ist doppelt so hoch wie die Laste in den Domänen A und E, weil die Domänen B, C und D jeweils einen Links zu zwei Domänen haben. Diese Verteilung der Last bliebe auch bei 1000 Domänen in der Reihe konstant, weil sich die Last nach der Anzahl der Links jeder Domäne richtet und nicht nach der Gesamtanzahl der Domänen in der Topologie.

Leistungsaspekte

Berücksichtigen Sie bei der Verwendung von Multimaster-Replikationstopologien die folgenden Einschränkungen:

- **Optimierung der Verteilung von Änderungen** (zuvor beschrieben)
- **Latenzzeit bei der Replikation** (zuvor beschrieben)
- **Leistung von Replikationslinks**: eXtreme Scaleerstellt einen einzigen TCP/IP-Socket zwischen jedem JVM-Paar. Der gesamte Datenverkehr zwischen diesen JVMs findet über diesen Socket statt, einschließlich der Multimaster-Replikation. Da sich Domänen in mindestens N Container-JVMs mit N TCP-Links zu Peer-Domänen befinden, weisen die Domänen mit einer höheren Anzahl an Containern höhere Replikationsleistungsraten auf. Mehr Container bedeuten mehr CPU- und Netzressourcen.
- **Optimierung des TCP-Sliding-Window und RFC 1323**: Wenn Sie die Unterstützung für RFC 1323 auf beiden Seiten eines Links aktivieren, werden mehr Daten in einer Austauschoperation zugelassen, was zu einem höheren Durchsatz führt. Die Technik erweitert die Fensterkapazität um einen Faktor von ca. 16.000.

  TCP-Sockets verwenden einen Sliding-Window-Mechanismus, um den Fluss von Massendaten zu steuern, der den Socket gewöhnlich auf 64 KB für ein Umlaufintervall beschränkt. Wenn ein Umlaufintervall 100 ms lang ist, ist die Bandbreite ohne Optimierung auf 640 KB/Sekunde beschränkt. Um die verfügbare Bandbreite eines Links vollständig nutzen zu können, müssen unter Umständen spezielle Optimierungs-Tasks für ein Betriebssystem ausgeführt werden. Die meisten Betriebssysteme haben Optimierungsparameter, einschließlich Optionen für RFC 1323, um den Durchsatz über Links mit hoher Latenzzeit zu verbessern.

  Es gibt mehrere Faktoren, die sich auf die Replikationsleistung auswirken können:

  – Geschwindigkeit, mit der eXtreme Scale Änderungen beziehen kann,
  – Geschwindigkeit, mit der eXtreme Scale Replikationsanforderungen bearbeiten kann,
  – Kapazität des Sliding Window,
  – Optimierung des Netzpuffers auf beiden Seiten eines Links, damit eXtreme Scale Änderungen über den Socket so schnell wie möglich beziehen kann.
- **Objektserialisierung**: Alle Daten müssen serialisierbar sein. Wenn eine Domäne COPY_TO_BYTES nicht verwendet, muss die Domäne Java-Serialisierung oder ObjectTransformer verwenden, um die Serialisierungsleistung zu optimieren.
- **Komprimierung**: eXtreme Scale komprimiert standardmäßig alle Daten, die zwischen Domänen gesendet werden. Es gibt keine Option für die Inaktivierung der Komprimierung im aktuellen Release.
- **Hauptspeicheroptimierung**: *Die Speicherbelegung für eine Multimaster-Replikationstopologie ist weitgehend unabhängig von der Anzahl der Domänen in der Topologie.*

  Die Aktivierung der Multimaster-Replikation bedeutet feste Gemeinkosten pro Map-Eintrag für die Versionssteuerung. Außerdem überwacht jeder Container ein festes Datenvolumen für jede Domäne in der Topologie. Eine Topologie mit zwei Domänen belegt ungefähr denselben Speicher wie eine Topologie mit 50 Domänen. eXtreme Scale verwendet keine Wiedergabeprotokolle oder ähnlichen Warteschlangen in der Implementierung, d. h., wenn ein Replikationslink für längere Zeit nicht verfügbar ist, gibt Datenstruktur mit zunehmender Größe, die darauf wartet, die Replikation fortzusetzen, wenn der Link erneut gestartet wird.

Mehrere Rechenzentren mit FIXED_PARTITION

Sie können jetzt ein FIXED_PARTITION-Grid zwischen zwei oder mehr Rechenzentren einsetzen. Jedes Rechenzentrum benötigt im Hinblick auf die Multimaster-Replikation eine eigene Domäne. Jedes Rechenzentrum kann Daten aus der lokalen Domäne lesen und in diese schreiben. Diese Änderungen werden an die anderen Rechenzentren über die von Ihnen definierten Links weitergegeben.

Vollständig replizierte Clients

Diese Topologievariante umfasst ein eXtreme-Scale-Serverpaar, das als Hub ausgeführt wird. Jeder Client erstellt ein eigenständiges Einzelcontainer-Grid mit einem Katalog in der Client-JVM. Ein Client verwendet sein Grid, um die Verbindung zum Hub-Katalog herzustellen, was bewirkt, dass sich der Client mit dem Hub synchronisiert, sobald die Verbindung zum Hub hergestellt ist.

Alle vom Client vorgenommenen Änderungen sind lokal und werden asynchron im Hub repliziert. Der Hub dient als Arbitrierungsdomäne und verteilt Änderungen an alle verbundenen Clients. Die Topologie mit vollständig replizierten Clients ist ein guter L2-Cache für einen objektbezogenen Mapper wie OpenJPA. Änderungen werden über den Hub schnell an die Client-JVMs verteilt. Solange die Cachegröße vom verfügbaren Heap-Speicher der Clients untergebracht werden kann, ist diese Topologie eine geeignete Architektur für diesen L2-Stil.

Verwenden Sie bei Bedarf mehrere Partitionen für die Skalierung der Hub-Domäne in mehreren JVMs. Da alle Daten weiterhin in eine einzige Client-JVM passen müssen, erhöht die Verwendung mehrerer Partitionen die Kapazität des Hubs für die Verteilung und Arbitrierung von Änderungen, aber nicht die Kapazität einer einzelnen Domäne.

Einschränkungen

Berücksichtigen Sie bei der Entscheidung, ob und wie Multimaster-Replikationstopologien zu verwenden sind, die folgenden Einschränkungen:

- **Konfiguration von Loadern mit mehreren Domänen**

  Domänen müssen Zugriff auf alle Klassen haben, die als Schlüssel und Werte verwendet werden. Alle Abhängigkeiten müssen sich in allen Klassenpfaden für Grid-Container-JVMs für alle Domänen widerspiegeln. Wenn ein CollisionArbiter-Plug-in den Wert für einen Cacheeintrag abruft, müssen die Klassen für die Werte für die Domäne vorhanden sein, die den Arbiter aufruft.
- **Verwendung von Loadern wird nicht empfohlen**

  Loader können verwendet werden, um Änderungen zwischen einem Grid und einer Datenbank zu übertragen. Es ist unwahrscheinlich, dass alle Grids (Domänen) in einer Topologie geografisch mit derselben Datenbank kollokiert sind. Aufgrund der Latenzzeit im WAN und anderen Faktoren kann dieser Anwendungsfall unerwünscht sein.

  Das vorherige Laden (Preload) des Grids ist ein weiterer Punkt, der eines sorgfältigen Designs bedarf. Beim Neustart eines Grids wird der Preload gewöhnlich erneut durchgeführt. Ein Preload ist bei der Verwendung einer Multimaster-Replikation nicht erforderlich oder sogar nicht erwünscht. Sobald eine Domäne online ist, lädt sie automatisch den Inhalt der Domänen, mit denen sie verlinkt ist. Deshalb ist es nicht erforderlich, einen manuellen Preload für ein Grid einzuleiten, das eine Domäne in einer Multimaster-Replikationstopologie ist.

Loader halten gewöhnlich Einfüge- und Aktualisierungsregeln ein. Bei der Multimaster-Replikation müssen Einfügungen als Zusammenführungen behandelt werden. Wenn die Daten nach einem Domänenneustart über Fernzugriff abgerufen werden, werden vorhandene Daten in die lokale Domäne "eingefügt". Da diese Daten unter Umständen bereits in der lokalen Datenbank enthalten sind, schlägt eine typische Einfügung mit einer Ausnahme wegen doppelt vorhandener Schlüssel in der Datenbank fehl. Stattdessen muss die Semantik für Zusammenführung (merge) verwendet werden.

eXtreme Scale kann für die Durchführung eines Shard-basierten Preloads unter Verwendung der Preload-Methoden in Loader-Plug-ins konfiguriert werden. Verwenden Sie diese Technik nicht in einer Multimaster-Replikationstopologie. Verwenden Sie stattdessen einen clientbasierte Preload, wenn die Topologie (anfänglich) gestartet wird. Lassen Sie zu, dass die Multimaster-Topologie alle neu gestarteten Domänen mit einer aktuellen Kopie der in anderen Domänen der Topologie gespeicherten Daten aktualisiert. Nach dem Start von Domänen ist die Multimaster-Topologie für die Synchronisation der Domänen zuständig.

- **Keine Unterstützung für EntityManager**

  Ein MapSet, das eine Entitäts-Map enthält, wird in Domänen nicht repliziert.
- **Keine Unterstützung für Bytefeldgruppen-Maps**

  Ein MapSet, das eine Map enthält, die mit COPY_TO_BYTES konfiguriert ist, wird in Domänen nicht repliziert.
- **Keine Unterstützung für Write-behind-Operationen**

  Ein MapSet, das eine Map enthält, die mit Write-behind-Unterstützung konfiguriert ist, wird in Domänen nicht repliziert.

Implementierungskonfigurationen für eXtreme Scale

WebSphere eXtreme Scale kann in zwei Typen von Umgebungen implementiert werden: lokalen und verteilten. Die erforderliche Konfiguration richtet sich nach dem Typ der Implementierungsumgebung.

Lokale Umgebungen

Wenn Sie das Produkt in einer lokalen Umgebung implementieren, wird eXtreme Scale in einer einzigen Java Virtual Machine ausgeführt und wird nicht repliziert. Für eine lokale Umgebung benötigen Sie eine ObjectGrid-XML-Datei oder ObjectGrid-APIs.

Verteilte Umgebungen

Wenn Sie das Produkt in einer verteilten Umgebung implementieren, wird eXtreme Scale in mehreren Java Virtual Machines ausgeführt. Mit dieser Konfiguration können Sie Datenreplikation und Partitionierung verwenden. Eine verteilte Umgebung kann noch weiter als dynamische Implementierungstopologie klassifiziert werden, in der Sie Server nach Bedarf hinzufügen können. Wie bei einer lokalen Umgebung ist in einer verteilten Umgebung eine ObjectGrid-XML-Datei oder eine entsprechende programmgesteuerte Konfiguration erforderlich. Außerdem müssen Sie eine XML-Datei für Implementierungsrichtlinien mit Konfigurationsdetails für Ihr speicherinternes eXtreme-Scale-Daten-Grid bereitstellen.

Katalogservice mit hoher Verfügbarkeit

Eine Katalogservicedomäne ist das Grid der verwendeten Katalogserver, die Topologieinformationen für alle Container in Ihrer eXtreme-Scale-Umgebung enthalten. Der Katalogservice steuert den Lastausgleich und das Routing für alle Clients. Wenn Sie eXtreme Scale als speicherinternen Verarbeitungsbereich implementieren möchten, müssen Sie den Katalogservice für die hohe Verfügbarkeit zu einer Katalogservicedomäne zusammenfassen.

Komponenten der Katalogservicedomäne

Wenn mehrere Katalogserver gestartet werden, wird einer der Server als Masterkatalogserver ausgewählt. Dieser Masterserver akzeptiert IIOP-Überwachungssignale (Internet Inter-ORB Protocol) und bearbeitet Systemdatenänderungen, die sich aufgrund von Änderungen im Katalogservice oder in den Containern ergeben.

Wenn Clients einen Kontakt zu einem der Katalogserver herstellen, wird die Routing-Tabelle für die Katalogservicedomäne über den CORBA-Servicekontext (Common Object Request Broker Architecture) an die Clients weitergegeben.

Konfigurieren Sie mindestens drei Katalogserver. Wenn Ihre Konfiguration Zonen enthält, können Sie einen Katalogserver pro Zone konfigurieren.

Wenn ein Server und ein Container von eXtreme Scale den Kontakt zu einem der Katalogserver herstellen, wird die Routing-Tabelle für die Katalogservicedomäne ebenfalls über den CORBA-Servicekontext an den Server und Container von eXtreme Scale weitergegeben. Ist der kontaktierte Katalogserver derzeit nicht der Masterkatalogserver, wird die Anforderung automatisch an den aktuellen Masterkatalogserver umgeleitet und die Routing-Tabelle für den Katalogserver aktualisiert.

Anmerkung: Ein Katalogserver-Grid und ein Containerserver-Grid unterscheiden sich in vielerlei Hinsicht. Die Katalogservicedomäne ist für die hohe Verfügbarkeit Ihrer Systemdaten verantwortlich. Das Container-Grid ist für die hohe Verfügbarkeit, die Skalierbarkeit und das Workload-Management Ihrer Daten verantwortlich. Deshalb sind zwei verschiedene Routing-Tabellen vorhanden: die Routing-Tabelle für die Katalogservicedomäne und die Routing-Tabelle für die Server-Grid-Shards.

Die Zuständigkeiten des Katalogs sind in eine Reihe von Services unterteilt. Der Stammgruppenmanager führt die Peer-Gruppierung für die Vitalitätsüberwachung durch, der Verteilungsservice führt die Zuordnung durch, der Verwaltungsservice ermöglicht den Zugriff auf die Verwaltung, und der Positionsservice verwaltet die Positionen.

Implementierung der Katalogservicedomäne

Stammgruppenmanager

Der Katalogservice verwendet den High Availability Manager (kurz HA-Manager), um Prozesse für die Überwachung der Verfügbarkeit zu gruppieren. Jede Prozessgruppierung ist eine Stammgruppe. Mit eXtreme Scale gruppiert der Stammgruppenmanager die Prozesse dynamisch. Diese Prozesse werden für die Skalierbarkeit klein gehalten. Jede Stammgruppe wählt ein führendes Member aus, das zusätzlich dafür verantwortlich ist, Statusnachrichten an den Stammgruppenmanager zu senden, wenn einzelne Member ausfallen. Über denselben Statusmechanismus wird erkannt, wenn alle Member einer Gruppe ausfallen und daraufhin die Kommunikation mit dem führenden Member verloren geht.

Der Stammgruppenmanager ist ein vollständig automatischer Service, der für die Organisation der Container in kleine Servergruppen zuständig ist, die dann automatisch lose miteinander zu einem ObjectGrid verbunden werden. Wenn ein Container den ersten Kontakt zum Katalogservice herstellt, wartet er auf die Zuteilung einer neuen oder vorhandenen Gruppe. Eine Implementierung von eXtreme Scale setzt sich aus vielen solcher Gruppen zusammen, und diese Gruppierung ist ein Enabler für die Skalierbarkeit von Schlüsseln. Jede Gruppe ist eine Gruppe von Java Virtual Maschines, die den Austausch von Überwachungssignalen verwenden, um die Verfügbarkeit der jeweils anderen Gruppen zu überwachen. Eines dieser Gruppen-Member wird als führendes Member ausgewählt und ist zusätzlich dafür verantwortlich, die Verfügbarkeitsinformationen an den Katalogservice zu übermitteln, um durch Neuzuordnung und Routenweiterleitung auf Fehler reagieren zu können.

Verteilungsservice

Der Katalogservice verwaltet die Verteilung von Shards auf die verfügbaren Container. Der Verteilungsservice ist dafür verantwortlich, dass die Ressourcen gleichmäßig auf die physischen Ressourcen verteilt werden. Der Verteilungsservice ordnet die einzelnen Shards ihren Hostcontainern zu. Der Verteilungsservice wird als einer von N ausgewählten Services im Daten-Grid ausgeführt, so dass genau eine Instanz des Service aktiv ist. Wenn diese Instanz ausfällt, wird ein anderer Prozess ausgewählt, der die Arbeit dieser Instanz übernimmt. Aus Redundanzgründen wird der Status des Katalogservice in allen Servern, in denen der Katalogservice ausgeführt, repliziert.

Verwaltung

Der Katalogservice ist auch der logische Einstiegspunkt für die Systemverwaltung. Der Katalogservice enthält eine Managed Bean (MBean) und stellt JMX-URLs (Java Management Extensions) für alle vom Service verwalteten Server bereit.

Positionsservice

Der Positionsservice tritt als Touchpoint für Clients, die die Container suchen, in denen die gewünschte Anwendung ausgeführt wird, sowie für die Container auf, die in ihnen ausgeführte Anwendungen beim Verteilungsservice registrieren möchten. Der Positionsservice wird zur horizontalen Skalierung dieser Funktion in allen Grid-Membern ausgeführt.

Der Katalogservice enthält Logik, die in stabilen Zuständen gewöhnlich inaktiv ist. Deshalb wirkt sich der Katalogservice nur geringfügig auf die Skalierbarkeit aus. Der Service ist so konzipiert, dass er Hunderte von Containern bedienen kann, die gleichzeitig verfügbar werden. Für die Verfügbarkeit konfigurieren Sie den Katalogservice in einem Grid.

Planung

Nach dem Starten einer Katalogservicedomäne werden die Member des Grids miteinander verbunden. Planen Sie die Topologie Ihrer Katalogservicedomäne sorgfältig, weil die Konfiguration der Katalogservicedomäne zur Laufzeit nicht geändert werden kann. Verteilen Sie das Grid so breit wie möglich, um Fehler zu verhindern.

Katalogservicedomäne starten

Weitere Informationen zum Erstellen einer Katalogservicedomäne finden Sie in den Details zum Starten einer Katalogservicedomäne im *Administratorhandbuch*.

Container von eXtreme Scale, die in WebSphere Application Server integriert sind, zu einer eigenständigen Katalogservicedomäne verbinden

Sie können Container von eXtreme Scale, die in eine Umgebung von WebSphere Application Server integriert sind, zu einer eigenständigen Katalogservicedomäne verbinden.

- 7.1+ Sie können Katalogservicedomänen in der Administrationskonsole erstellen. Weitere Einzelheiten finden Sie in den Informationen zum Erstellen von Katalogservicedomänen im *Administratorhandbuch*.
- (Veraltet) In früheren Releases haben Sie die Katalogservices über eine angepasste Eigenschaft mit einer Katalogservicedomäne verbunden. Sie können diese Eigenschaft zwar weiterhin verwenden, aber sie ist veraltet. Weitere Einzelheiten zu dieser angepassten Eigenschaft finden Sie in den Informationen zum Starten des Katalogserviceprozesses in einem WebSphere Application Server im *Administratorhandbuch*.

  Anmerkung: Servernamenskollision: Da diese Eigenschaft sowohl zum Starten des Katalogservers von eXtreme Scale als auch zum Herstellen einer Verbindung zum Katalogserver verwendet wird, dürfen die Katalogserver keinen Namen haben, der mit einem der Server von WebSphere Application Server übereinstimmt.

Weitere Informationen finden Sie in den Informationen zu Katalogserver-Quorums in der *Produktübersicht*.

Katalogserver-Quorum

Das Quorum ist die minimale Anzahl an Katalogservern, die erforderlich ist, um Verteilungsoperationen für das Daten-Grid durchzuführen. Die minimale Anzahl ist die vollständige Gruppe der Katalogserver, sofern Sie das Quorum nicht überschreiben.

Wichtige Begriffe

Im Folgenden finden Sie eine Liste der Begriffe, die sich auf Quorumaspekte für WebSphere eXtreme Scale beziehen.

- **Brownout**: Ein Brownout ist der temporäre Verlust der Konnektivität zwischen Servern.
- **Blackout:** Ein Blackout ist der permanente Verlust der Konnektivität zwischen Servern.
- **Rechenzentrum:** Ein Rechenzentrum ist eine sich an einem bestimmten Standort befindliche Gruppe von Servern, die im Allgemeinen über ein lokales Netz (LAN, Local Area Netzwork) miteinander verbunden sind.
- **Zone:** Eine Zone ist eine Konfigurationsoption, die verwendet wird, um Server zu gruppieren, die ein gemeinsames physisches Merkmal haben. Zonenbeispiele für eine Gruppe von Servern sind ein Rechenzentrum, wie im vorherigen Listenpunkt beschrieben, ein Bereichsnetz, ein Gebäude, ein Stockwerk eines Gebäudes usw.
- **Überwachungssignal**: Der Austausch von Überwachungssignalen ist ein Mechanismus, der verwendet wird, um festzustellen, ob eine bestimmte Java Virtual Machine (JVM) aktiv ist.

Topologie

In diesem Abschnitt wird erläutert, wie WebSphere eXtreme Scale in einem Netz arbeitet, das unzuverlässige Komponenten enthält. Ein Beispiel für ein solches Netz ist ein Netz, das sich über mehrere Rechenzentren erstreckt.

IP-Adressraum

WebSphere eXtreme Scale erfordert ein Netz, in dem alle adressierbaren Elemente im Netz ungehindert eine Verbindung zu jedem anderen adressierbaren Element im Netz herstellen können. Dies bedeutet, dass WebSphere eXtreme Scale einen flachen IP-Adressraum erfordert und Firewalls für den gesamten Datenverkehr voraussetzt, der zwischen den IP-Adressen und Ports übertragen wird, die von den Java Virtual Machines (JVM) verwendet werden, die Elemente von WebSphere eXtreme Scale enthalten.

Verbundene LANs

Jedem lokalen Netz (LAN, Local Area Network) wird eine Zonenkennung für die Anforderungen von WebSphere eXtreme Scale zugeordnet. WebSphere eXtreme Scale fragt JVMs in einer einzigen Zone aggressiv mit Überwachungssignalen ab. Ein verpasstes Überwachungssignal löst ein Failover-Ereignis aus, wenn der Katalogservice das Quorum hat.

Katalogservicedomäne und Containerserver

Eine Katalogservicedomäne ist eine Sammlung ähnlicher JVMs. Eine Katalogservicedomäne ist ein Grid, das sich aus Katalogservern zusammensetzt und eine feste Größe hat. Die Anzahl der Containerserver ist jedoch dynamisch. Containerserver können nach Bedarf hinzugefügt und entfernt werden. In einer Konfiguration mit drei Rechenzentren erfordert WebSphere eXtreme Scale eine Katalogservice-JVM pro Rechenzentrum.

Die Katalogservicedomäne verwendet einen Mechanismus mit vollständigem Quorum. Das bedeutet, dass alle Member des Daten-Grids einer Aktion zustimmen müssen.

Containerserver-JVMs werden mit einer Zonenkennung gekennzeichnet. Das Grid der Container-JVMs wird automatisch in kleinere JVM-Stammgruppen aufgeteilt. Eine Stammgruppe enthält nur die JVMs aus einer einzigen Zone. JVMs unterschiedlicher Zonen werden nie in dieselbe Stammgruppe aufgenommen.

Eine Stammgruppe versucht aggressiv, Ausfälle ihrer Member-JVMs zu erkennen. Die Container-JVMs einer Stammgruppe dürfen sich nicht über mehrere LANs erstrecken, die über Verbindungen wie ein Weitverkehrsnetz (WAN, Wide Area Network) miteinander verbunden sind. Dies bedeutet, dass eine Stammgruppe keine Container aus derselben Zone enthalten kann, die in anderen Rechenzentren ausgeführt werden.

Serverlebenszyklus

Katalogserverstart

Die Katalogserver werden mit dem Befehl "startOgServer" gestartet. Der Quorum-Mechanismus ist standardmäßig inaktiviert. Zum Aktivieren des Quorums können Sie das Flag "`-quorum` enabled" an den Befehl "startOgServer" übergeben oder die

Eigenschaft "enableQuorum=true" in der Eigenschaftendatei hinzufügen. Für alle Katalogserver muss dieselbe Quorum-Einstellung festgelegt werden.

```
# bin/startOgServer cat0 –serverProps objectGridServer.properties
```

Datei "objectGridServer.properties"

```
catalogClusterEndPoints=cat0:cat0.domain.com:6600:6601,
cat1:cat1.domain.com:6600:6601
catalogServiceEndPoints= cat0.domain.com:2809, cat1.domain.com:2809
enableQuorum=true
```

Containerserverstart

Die Containerserver werden mit dem Befehl "startOgServer" gestartet. Wenn ein Daten-Grid über mehrere Rechenzentren verteilt ausgeführt wird, müssen die Server die Zonenkennung verwenden, um das Rechenzentrum zu identifizieren, in dem sie sich befinden. Die Einstellung der Zone in den Grid-Servern ermöglicht WebSphere eXtreme Scale, den Zustand der Server zu überwachen, die dem Rechenzentrum zugeordnet sind, und somit den rechenzentrumsübergreifenden Datenverkehr zu minimieren.

```
# bin/startOgServer gridA0 –serverProps objectGridServer.properties –
objectgridfile xml/objectgrid.xml –deploymentpolicyfile xml/
deploymentpolicy.xml
```

Datei "objectGridServer.properties"

```
catalogServiceEndPoints= cat0.domain.com:2809, cat1.domain.com:2809
zoneName=ZoneA
```

Grid-Server herunterfahren

Die Grid-Server werden mit dem Befehl "stopOgServer" gestoppt. Wenn Sie ein gesamtes Rechenzentrum für Wartungsarbeiten herunterfahren möchten, übergeben Sie die Liste aller Server, die zu dieser Zone gehören. Die Übergabe dieser Liste ermöglicht einen sauberen Statusübergang von der Zone, die umgerüstet wird, auf die noch aktiven Zonen.

```
# bin/stopOgServer gridA0,gridA1,gridA2 –catalogServiceEndPoints
cat0.domain.com:2809,cat1.domain.com:2809
```

Ausfallerkennung

WebSphere eXtreme Scale erkennt Prozessabstürze anhand abnormaler Beendigungsereignisse für Sockets. Der Katalogservice wird sofort darüber benachrichtigt, wenn ein Prozess beendet wird. Ein Blackout wird anhand von nicht beantworteten Überwachungssignalen erkannt. WebSphere eXtreme Scale schützt sich durch die Verwendung einer Quorum-Implementierung selbst vor Brownout-Bedingungen in Rechenzentren.

Implementierung des Austauschs von Überwachungssignalen

In diesem Abschnitt wird beschrieben, wie die Aktivitätsprüfung in WebSphere eXtreme Scale implementiert wird.

Austausch von Überwachungssignalen zwischen Stammgruppen-Membern

Der Katalogservice verteilt Container-JVMs auf Stammgruppen begrenzter Größe. Eine Stammgruppe verwendet zwei Methoden, um den Ausfall ihrer Member zu erkennen. Wenn der Socket einer JVM geschlossen wird, wird diese JVM als inak-

tiv eingestuft. Außerdem sendet jedes Member in einem in der Konfiguration festgelegten Intervall Überwachungssignale über diese Sockets. Wenn eine JVM nicht innerhalb der konfigurierten maximalen Zeitspanne auf diese Überwachungssignale antwortet, wird die JVM als inaktiv eingestuft.

Ein einziges Member jeder Stammgruppe wird als führendes Member bestimmt. Das führende Stammgruppen-Member ist dafür verantwortlich, dem Katalogservice in regelmäßigen Abständen den Aktivitätsstatus der Stammgruppe und alle Änderungen der Stammgruppenzugehörigkeiten zu melden. Eine Zugehörigkeitsänderung kann beispielsweise der Ausfall einer JVM oder das Hinzufügen einer neuen JVM sein, die in die Stammgruppe aufgenommen wird.

Wenn das führende Stammgruppen-Member zu keinem der Member der Katalogservicedomäne eine Verbindung herstellen kann, setzt es seine Verbindungsversuche fort.

Austausch von Überwachungssignalen in der Katalogservicedomäne

Die Katalogservicedomäne gleicht einer privaten Stammgruppe mit einer statischen Zugehörigkeit und einem Quorum-Mechanismus. Sie erkennt Ausfälle auf dieselbe Weise wie eine normale Stammgruppe. Das Verhalten ist jedoch anders und umfasst eine Quorum-Logik. Außerdem verwendet der Katalogservice eine weniger aggressive Konfiguration für den Austausch von Überwachungssignalen.

Austausch von Überwachungssignalen in einer Stammgruppe

Der Katalogservice muss über den Ausfall von Containerservern informiert werden. Jede Stammgruppe ist für die Erkennung des Ausfalls der Container-JVM und die Meldung dieses Ausfalls an den Katalogservice durch das führende Stammgruppen-Member verantwortlich. Der vollständige Ausfall aller Member einer Stammgruppe ist auch möglich. Wenn die vollständige Stammgruppe ausfällt, muss der Katalogservice diesen Verlust erkennen.

Wenn der Katalogservice eine Container-JVM als ausgefallen kennzeichnet und der Container später wieder als aktiv gemeldet wird, wird die Container-JVM angewiesen, die Containerserver von WebSphere eXtreme Scale herunterzufahren. Eine JVM in diesem Status ist in Abfragen mit dem Befehl "xsadmin" nicht sichtbar. Die Nachrichten in den Protokollen der Container-JVM weisen darauf hin, dass die Container-JVM ausgefallen ist. Sie müssen diese JVM manuell erneut starten.

Bei einem Verlust des Quorums wird der Austausch von Überwachungssignalen so lange ausgesetzt, bis das Quorum wiederhergestellt ist.

Quorum-Verhalten des Katalogservice

Normalerweise haben die Member des Katalogservice vollständige Konnektivität. Die Katalogservicedomäne ist eine statische Gruppe von JVMs. WebSphere eXtreme Scale erwartet, dass stets alle Member des Katalogservice online sind. Der Katalogservice reagiert nur auf Containerereignisse, wenn der Katalogservice das Quorum hat.

Verliert der Katalogservice das Quorum, wartet er, bis das Quorum wiederhergestellt ist. In der Zeit, in der der Katalogservice kein Quorum hat, ignoriert er alle Ereignisse der Containerserver. Containerserver wiederholen alle Anforderungen, die vom Katalogserver in der Zeit zurückgewiesen werden, da WebSphere eXtreme Scale von der Wiederherstellung des Quorums ausgeht.

Die folgende Nachricht zeigt einen Quorum-Verlust an. Suchen Sie nach dieser Nachricht in Ihren Katalogserviceprotokollen.

```
CWOBJ1254W: Der Katalogservice wartet auf das Quorum.
```

WebSphere eXtreme Scale erwartet in den folgenden Fällen einen Quorum-Verlust:

- Das Member mit der Katalogservice-JVM fällt aus.
- Es tritt ein Netz-Brownout ein.
- Es tritt ein Ausfall des Rechenzentrums ein.

Das Stoppen einer Katalogserverinstanz mit dem Befehl `stopOgServer` führt nicht zum Verlust des Quorums, weil das System weiß, dass die Serverinstanz gestoppt wurde. Dies ist etwas anderes als ein JVM-Ausfall oder ein Brownout.

Quorum-Verlust nach JVM-Ausfall

Ein Katalogserver, der ausfällt, führt zu einem Quorum-Verlust. In diesem Fall muss das Quorum so schnell wie möglich wiederhergestellt werden. Ein ausgefallener Katalogservice kann dem Grid erst dann wieder beitreten, wenn das Quorum wiederhergestellt ist.

Quorum-Verlust nach einem Netz-Brownout

WebSphere eXtreme Scale ist auf die Möglichkeit von Brownouts vorbereitet. Ein Brownout ist ein temporärer Verlust der Konnektivität zwischen Rechenzentren. Brownout-Bedingungen sind gewöhnlich transient und sollten innerhalb von Sekunden bzw. Minuten behoben sein. WebSphere eXtreme Scale versucht während eines Brownouts den normalen Betrieb aufrecht zu erhalten. Ein Brownout wird als einzelnes Fehlereignis betrachtet. Es wird davon ausgegangen, dass der Fehler behoben und der normale Betrieb anschließend fortgesetzt wird, ohne dass Aktionen von WebSphere eXtreme Scale erforderlich sind.

Ein langer Brownout kann nur durch Benutzereingriff als Blackout klassifiziert werden. Das Quorum muss auf einer Seite des Brownouts wiederhergestellt werden, damit das Ereignis als als Blackout klassifiziert werden kann.

JVM des Katalogservice stoppen und erneut starten

Wenn ein Katalogserver mit dem Befehl "stopOgServer" gestoppt wird, sinkt das Quorum um einen Server. Das bedeutet, dass die verbleibenden Server weiterhin das Quorum haben. Bei einem Neustart des Katalogservers steigt das Quorum wieder auf die verherige Zahl.

Konsequenzen eines Quorum-Verlusts

Wenn eine Container-JVM ausfällt und das Quorum verloren gegangen ist, findet erst dann eine Wiederherstellung statt, wenn die Brownout-Bedingung behoben ist bzw. wenn der Kunde im Fall eines Blackouts einen Befehl zum Wiederherstellen des Quorums absetzt. WebSphere eXtreme Scale betrachtet einen Quorum-Verlust und einen Containerausfall als doppelten Fehler, aber ein solches Ereignis tritt nur selten ein. Das bedeutet, dass Anwendungen den Schreibzugriff auf Daten, die in der ausgefallenen JVM gespeichert wurden, so lange verlieren können, bis das Quorum wiederhergestellt ist und die normale Wiederherstellung stattfindet.

Wenn Sie versuchen, einen Container zu starten, während das Quorum verloren ist, wird der Container nicht gestartet.

Während eines Quorum-Verlusts ist eine vollständige Clientkonnektivität zulässig. Wenn während des Quorum-Verlusts keine Containerausfälle oder Konnektivitätsprobleme auftreten, können die Clients weiterhin vollständig mit den Containerservern interagieren.

Wenn ein Brownout auftritt, haben einige Clients möglicherweise erst dann wieder Zugriff auf die primären Kopieren oder Replikatkopien der Daten, wenn die Brownout-Bedingung behoben ist.

Neue Clients können gestartet werden, da in jedem Rechenzentrum eine Katalogservice-JVM vorhanden sein sollte, so dass der Client selbst bei einem Brownout-Ereignis mindestens eine Katalogservice-JVM erreichen kann.

Wiederherstellung des Quorums

Wenn das Quorum aus irgendeinem Grund verloren geht, wird zur Wiederherstellung des Quorums ein Wiederherstellungsprotokoll ausgeführt. Beim Verlust des Quorums wird die Aktivitätsprüfung für die Stammgruppen ausgesetzt, und alle Ausfallberichte werden ignoriert. Sobald das Quorum wiederhergestellt ist, nimmt der Katalogservice die Aktivitätsprüfung aller Stammgruppen wieder auf, um ihre Zugehörigkeit unverzüglich zu bestimmen. Alle zuvor in den als ausgefallen gemeldeten Container-JVMs befindichen Shards werden wiederhergestellt. Wenn primäre Shards verloren gegangen sind, werden die noch aktiven Replikate zu primären Shards hochgestuft. Wenn Replikat-Shards verloren gegangen sind, werden zusätzliche Replikate in den noch aktiven JVMs erstellt.

Quorum überschreiben

Diese Option sollte nur verwendet werden, wenn ein Rechenzentrum ausfällt. Der Quorum-Verlust aufgrund des Ausfalls einer Containerservice-JVM oder eines Netz-Brownouts wird gewöhnlich automatisch behoben, sobald die Katalogservice-JVM erneut gestartet bzw. das Netz-Brownout behoben ist.

Nur Administratoren haben Kenntnis von einem Ausfall eines Rechenzentrums. WebSphere eXtreme Scale behandelt Brownouts und Blackouts ähnlich. Sie müssen die eXtreme-Scale-Umgebung über solche Ausfälle mit dem Befehl "xsadmin" in Kenntnis setzen, um das Quorum zu überschreiben. Auf diese Weise wird dem Katalogservice mitgeteilt, davon auszugehen, dass das Quorum mit den aktuellen Zugehörigkeiten erreicht ist und eine vollständige Wiederherstellung stattfindet. Wenn Sie einen Befehl zum Überschreiben des Quorums absetzen, bestätigen Sie, dass die JVMs im ausgefallenen Rechenzentrum wirklich ausgefallen sind und nicht wiederhergestellt werden.

In der folgenden Liste sind einige Szenarien für das Überschreiben des Quorums beschrieben. Angenommen, Sie haben drei Katalogserver: A, B, und C.

- Brownout: Angenommen, es tritt ein Brownout auf, bei dem C vorübergehend isoliert wird. Der Katalogservice verliert das Quorum und wartet auf die Behebung des Brownouts, nach der C der Katalogservicedomäne wieder beitritt und das Quorum wiederhergestellt ist. Ihre Anwendung stellt während dieser Zeit keine Probleme fest.
- Vorübergehender Ausfall: Hier fällt C aus, und der Katalogservice verliert das Quorum. In diesem Fall müssen Sie das Quorum überschreiben. Sobald das

Quorum wiederhergestellt ist, kann C erneut gestartet werden. C tritt der Katalogservicedomäne nach dem Neustart wieder bei. Die Anwendung stellt während dieser Zeit keine Probleme fest.

- Ausfall eines Rechenzentrums: Sie verifizieren, dass das Rechenzentrum wirklich ausgefallen und vom Netz isoliert ist. Anschließend setzen Sie den Befehl "xsadmin" zum Überschreiben des Quorums ab. Die anderen beiden noch aktiven Rechenzentren führen eine vollständige Wiederherstellung durch, indem sie Shards, die sich im ausgefallenen Rechenzentrum befinden, ersetzen. Der Katalogservice wird jetzt mit einem vollständigen Quorum von A und B ausgeführt. Die Anwendung kann zwischen dem Beginn des Blackouts und dem Überschreiben des Quorums Verzögerungen oder Ausnahmen feststellen. Sobald das Quorum überschrieben ist, wird das Grid wiederhergestellt und der normale Betrieb fortgesetzt.
- Wiederherstellung des Rechenzentrums: Die noch aktiven Rechenzentren werden bereits mit überschriebenem Quorum ausgeführt. Wenn das Rechenzentrum, in dem C enthalten ist, erneut gestartet wird, müssen alle JVMs im Rechenzentrum erneut gestartet werden. Anschließend tritt C der vorhandenen Katalogservicedomäne wieder bei, und das Quorum wird ohne Benutzereingriff wiederhergestellt.
- Ausfall eines Rechenzentrums und Brownout: Das Rechenzentrum, in dem C enthalten ist, fällt aus. Das Quorum wird überschrieben und in den verbleibenden Rechenzentren wiederhergestellt. Wenn ein Brownout zwischen A und B auftritt, kommen die herkömmlichen Regeln für eine Wiederherstellung nach einem Brownout zur Anwendung. Nach der Behebung des Brownouts wird das Quorum wiederhergestellt, und die erforderliche Wiederherstellung nach einem Quorum-Verlust findet statt.

Containerverhalten

In diesem Abschnitt wird beschrieben, wie sich die Containerserver-JVMs verhalten, wenn das Quorum verloren geht und anschließend wiederhergestellt wird.

Container enthalten einen oder mehrere Shards. Shards sind primäre Kopien oder Replikatkopien für eine bestimmte Partition. Der Katalogservice ordnet Shards einem Container zu, und der Container berücksichtigt diese Zuordnung so lange, bis er neue Anweisungen vom Katalogservice erhält. Wenn ein primäres Shard in einem Container nicht mit einem Replikat-Shard kommunizieren kann, weil ein Brownout eingetreten ist, setzt es seine Kommunikationsversuche so lange fort, bis es neue Anweisungen vom Katalogservice erhält.

Wenn ein Netz-Brownout auftritt und ein primäres Shard nicht mehr mit dem Replikat kommunizieren kann, wiederholt es die Verbindung so lange, bis der Katalogservice neue Anweisungen bereitstellt.

Verhalten synchroner Replikate

Wenn eine Verbindung unterbrochen ist, kann das primäre Shard neue Transaktionen akzeptieren, solange mindestens so viele Replikate online sind, wie mit der Eigenschaft "minsync" für das Mapset festgelegt wurde. Wenn neue Transaktionen im primären Shard verarbeitet werden, während die Verbindung zum synchronen Replikat unterbrochen ist, wird der Replikatinhalt gelöscht und nach Wiederherstellung der Verbindung mit dem aktuellen Status des primären Shards synchronisiert.

Von der synchronen Replikation zwischen Rechenzentren und der synchronen Replikation über WAN-Verbindungen wird dringend abgeraten.

Verhalten asynchroner Replikate

Wenn eine Verbindung unterbrochen ist, kann das primäre Shard neue Transaktionen akzeptieren. Das primäre Shard puffert die Änderungen bis zu einem bestimmten Grenzwert. Wenn die Verbindung mit dem Replikat wiederhergestellt wird, bevor der Grenzwert erreicht ist, wird das Replikat mit den gepufferten Änderungen aktualisiert. Beim Erreichen des Grenzwerts löscht das primäre Shard die gepufferte Liste, und bei der Wiederherstellung der Verbindung zum Replikat wird der Replikatinhalt gelöscht und neu mit dem primären Shard synchronisiert.

Clientverhalten

Unabhängig davon, ob die Katalogservicedomäne das Quorum hat oder nicht, können Clients für Bootstrapping immer eine Verbindung zum Katalogserver herstellen. Der Client versucht, eine Verbindung zu einer Katalogserverinstanz herzustellen, um eine Routentabelle anzufordern und anschließend mit dem Grid zu interagieren. Die Netzkonnektivität kann die Interaktion des Clients mit einigen Partitionen aufgrund der Netzkonfiguration verhindern. Der Client kann für den Abruf ferner Daten eine Verbindung zu lokalen Replikaten herstellen, sofern er entsprechend konfiguriert ist. Clients können keine Daten aktualisieren, wenn die primäre Partition für diese Daten nicht verfügbar ist.

Quorum-Befehle mit xsadmin

In diesem Abschnitt werden für Quorum-Situationen hilfreiche xsadmin-Befehle beschrieben.

Quorum-Status abfragen

Der Quorum-Status einer Katalogserverinstanz kann mit dem Befehl "xsadmin" abgefragt werden.

```
xsadmin –ch cathost –p 1099 –quorumstatus
```

Es gibt fünf mögliche Ergebnisse.

- Quorum ist inaktiviert: Die Katalogserver werden in einem Modus ausgeführt, in dem das Quorum inaktiviert ist. Dieser Modus ist für Entwicklungsumgebungen und Umgebungen mit einem einzigen Rechenzentrum bestimmt. Er wird für Konfigurationen mit mehreren Rechenzentren nicht empfohlen.
- Quorum ist aktiviert, und der Katalogserver hat das Quorum: Das Quorum ist aktiviert, und das System arbeitet normal.
- Quorum ist aktiviert, aber der Katalogserver wartet auf das Quorum: Das Quorum ist aktiviert, und das Quorum ist verloren gegangen.
- Quorum ist aktiviert, und das Quorum wurde überschrieben: Das Quorum ist aktiviert, und das Quorum wurde überschrieben.
- Quorum-Status ist unzulässig: Wenn ein Brownout auftritt, wird der Katalogservice in zwei Partitionen, A und B, aufgeteilt. Der Katalogserver A hat das Quorum überschrieben. Die Netzpartition wird aufgelöst. Der Status des Servers in der Partition B ist unzulässig, und es ist ein Neustart der JVM erforderlich. Dieser Fall tritt auch ein, wenn die Katalog-JVM in Partition B wegen des Brownouts erneut gestartet wird und die Brownout-Bedingung anschließend behoben wird.

Quorum überschreiben

Der Befehl "xsadmin" kann zum Überschreiben des Quorums verwendet werden. Alle noch aktiven Katalogserverinstanzen können verwendet werden. Alle noch aktiven Instanzen werden benachrichtigt, wenn eine Instanz angewiesen wird, das Quorum zu überschreiben. Die Syntax ist wie folgt:

```
xsadmin –ch cathost –p 1099 –overridequorum
```

Diagnosebefehle

- Quorum-Status: Siehe die Beschreibung im vorherigen Abschnitt.
- Stammgruppen auflisten: Zeigt eine Liste aller Stammgruppen an. Die Member und führenden Member der Stammgruppen werden angezeigt.

  ```
  xsadmin –ch cathost –p 1099 –coregroups
  ```

- Server entfernen: Dieser Befehl entfernt einen Server manuell aus dem Grid. Dies ist normalerweise nicht erforderlich, da Server automatisch entfernt werden, wenn sie als ausgefallen erkannt werden, aber der Befehl wird für die Verwendung unter Anleitung der IBM Unterstützungsfunktion bereitgestellt.

  ```
  xsadmin –ch cathost –p 1099 –g Grid –teardown server1,server2,server3
  ```

- Routentabelle anzeigen: Dieser Befehl zeigt die aktuelle Routentabelle an, indem eine neue Clientverbindung zum Grid simuliert wird. Außerdem validiert der Befehl die Routentabelle, indem er prüft, ob alle Containerserver ihre Rolle in der Routentabelle kennen, z. B. welcher Typ von Shard für welche Partition bestimmt ist.

  ```
  xsadmin –ch cathost –p 1099 –g myGrid -routetable
  ```

- Nicht zugeordnete Shards anzeigen: Wenn einige Shards nicht im Grid verteilt werden können, können diese Shards mit diesem Befehl aufgelistet werden. Dies geschieht nur, wenn der Verteilungsservice eine Einschränkung hat, die die Verteilung verhindert. Wenn Sie beispielsweise JVMs auf einem einzelnen physischen System starten, das im Produktionsmodus arbeitet, können nur primäre Shards verteilt werden. Replikate werden nicht zugordnet, solange JVMs auf dem zweiten System gestartet werden. Der Verteilungsservice verteilt Replikate nur an JVMs, die andere Adressen haben als die JVMs, in denen sich die primären Shards befinden. Wenn eine Zone keine JVMs enthält, kann dies auch dazu führen, dass Shards nicht zugeordnet werden.

  ```
  xsadmin –ch cathost –p 1099 –g myGrid –unassigned
  ```

- Trace-Einstellungen festlegen: Dieser Befehl legt die Trace-Einstellungen für alle JVMs fest, die dem für den Befehl "xsadmin" angegebenen Filter entsprechen. Diese Einstellung ändert die Trace-Einstellungen nur so lange, bis ein anderer Befehl verwendet wird bzw. die geänderten JVMs ausfallen oder gestoppt werden.

  ```
  xsadmin –ch cathost –p 1099 –g myGrid –fh host1 –settracespec
  ObjectGrid*=event=enabled
  ```

  Dieser Befehl aktiviert die Trace-Erstellung für alle JVMs auf dem System mit dem angegebenen Hostnamen, in diesem Fall host1.

- Map-Größen prüfen: Der Befehl "mapsizes" ist hilfreich, um sicherzustellen, dass die Schlüsselverteilung auf die Shards im Schlüssel einheitlich erfolgt. Wenn einige Container erheblich mehr Schlüssel als andere haben, verwendet die Hash-Funktion für die Schlüsselobjekte wahrscheinlich eine schlechte Verteilung.

  ```
  xsadmin -ch cathost -p 1099 -g myGrid -m myMapSet -mapsizes myMap
  ```

Hinweise zur Transportsicherheit

Da Rechenzentren gewöhnlich auf verschiedene geografische Standorte verteilt sind, können Benutzer aus Sicherheitsgründen die Transportsicherheit zwischen den Rechenzentren aktivieren.

Lesen Sie die Informationen zur Sicherheit auf Transportschicht im *Administratorhandbuch*.

Prüfliste für die Betriebsbereitschaft

Verwenden Sie die Prüfliste für die Betriebsbereitschaft, um Ihre Umgebung für die Implementierung von WebSphere eXtreme Scale vorzubereiten.

Tabelle 6. Prüfliste für die Betriebsbereitschaft

Prüflistenpunkt	Weiterführende Informationen

Wenn Sie AIX verwenden, optimieren Sie die folgenden Betriebssystemeinstellungen:
TCP_KEEPINTVL
Die Einstellung TCP_KEEPINTVL gehört zu einem Socket-Keepalive-Protokoll, das die Erkennung von Netzausfällen ermöglicht. Die Eigenschaft gibt das Intervall an, in dem Pakete zum Validieren der Verbindung gesendet werden. Wenn Sie WebSphere eXtreme Scale verwenden, setzen Sie die Eigenschaft auf `10`. Zum Überprüfen der aktuellen Einstellung führen Sie den folgenden Befehl aus:
`# no –o tcp_keepintvl`
Führen Sie den folgenden Befehl aus, um die aktuelle Einstellung zu ändern:
`# no –o tcp_keepintvl=10`
Der Wert für die Einstellung TCP_KEEPINTVL wird in Halbsekunden angegeben.
TCP_KEEPINIT
Die Einstellung TCP_KEEPINIT gehört zu einem Socket-Keepalive-Protokoll, das die Erkennung von Netzausfällen ermöglicht. Die Eigenschaft gibt das Anfangszeitlimit für die TCP-Verbindung an. Wenn Sie WebSphere eXtreme Scale verwenden, setzen Sie die Eigenschaft auf `40`. Zum Überprüfen der aktuellen Einstellung führen Sie den folgenden Befehl aus:
`# no –o tcp_keepinit`
Führen Sie den folgenden Befehl aus, um die aktuelle Einstellung zu ändern:
`# no –o tcp_keepinit=40`
Der Wert für die Einstellung TCP_KEEPINIT wird in Halbsekunden angegeben.
	• Informationen zur Optimierung von AIX finden Sie unter "AIX-Systeme optimieren".

Aktualisieren Sie die Datei `orb.properties`, um das Transportverhalten des Grids zu ändern. Sie finden die Datei `orb.properties` im Verzeichnis `java/jre/lib`.	„ORB-Eigenschaftendatei” auf Seite 197

Tabelle 6. Prüfliste für die Betriebsbereitschaft (Forts.)

Prüflistenpunkt	Weiterführende Informationen
Verwenden Sie Parameter im Script "startOgServer", insbesondere die folgenden: • Legen Sie die Einstellungen für den Heap-Speicher mit dem Parameter **-jvmArgs** fest. • Legen Sie den Anwendungsklassenpfad und Anwendungseigenschaften mit dem Parameter **-jvmArgs** fest. • Setzen Sie den Parameter **-jvmArgs** für die Konfiguration der Agentenüberwachung. **Porteinstellungen** WebSphere eXtreme Scale muss für einige Transporte Kommunikationsports öffnen. Diese Ports werden alle dynamisch definiert. Wenn jedoch eine Firewall zwischen den Containern verwendet wird, müssen Sie die Ports angeben. Verwenden Sie die folgenden Portinformationen: **Listener-Port** Sie können das Argument **-listenerPort** verwenden, um den Port anzugeben, der für die Kommunikation zwischen Prozessen verwendet wird. **Stammgruppenport** Sie können das Argument **-haManagerPort** verwenden, um den Port anzugeben, der für die Fehlererkennung verwendet wird. Dieses Argument ist mit "peerPort" identisch. Beachten Sie, dass Stammgruppen nicht zonenübergreifend kommunizieren müssen. Deshalb müssen Sie diesen Port unter Umständen nicht setzen, wenn die Firewall für alle Member einer einzigen Zone offen ist. **JMX-Serviceport** Sie können das Argument **-JMXServicePort** verwenden, um den Port anzugeben, den der JMX-Service verwenden soll. **SSL-Port** Wenn Sie `-Dcom.ibm.CSI.SSLPort=1234` als Argument mit **-jvmArgs** angeben, wird der SSL-Port auf `1234` gesetzt. Der SSL-Port ist der sichere Port-Peer zum Listener-Port. **Clientport** Wird nur im Katalogservice verwendet. Sie können diesen Wert mit dem Argument **-catalogServiceEndPoints** angeben. Der Wert für diesen Parameter muss im folgenden Format angegeben werden: `Servername:Hostname:Clientport:Peerport`	„Script "startOgServer"" auf Seite 340
Stellen Sie sicher, dass die Sicherheitseinstellungen ordnungsgemäß konfiguriert sind: • Transport (SSL) • Anwendung (Authentifizierung und Berechtigung) Zum Überprüfen Ihrer Sicherheitseinstellungen können Sie versuchen, einen zerstörerischen Client zu verwenden, der eine Verbindung zu Ihrer Konfiguration herstellt. Wenn beispielsweise die Einstellung "SSL-Required" konfiguriert ist, sollte ein Client, der die Einstellung TCP_IP hat, oder ein Client mit dem falschen Truststore nicht in der Lage sein, eine Verbindung zum Server herzustellen. Wenn eine Authentifizierung erforderlich ist, sollte ein Client ohne Berechtigungsnachweis (z. B. ohne Benutzer-ID und Kennwort) nicht in der Lage sein, eine Verbindung zum Server herzustellen. Wenn die Berechtigung zwingend erforderlich ist, sollte einem Client ohne Zugriffsberechtigung der Zugriff auf die Serverressourcen nicht erteilt werden.	„Sicherheitsintegration mit externen Providern" auf Seite 375
Legen Sie fest, wie die Umgebung überwacht werden soll. • xsAdmin – Die JMX-Ports der Katalogserver müssen für das Tool "XSAdmin" sichtbar sein. Auch die Containerport müssen für einige Befehle, die Informationen von den Containern erfassen, zugänglich sein. • Sie können zwischen den folgenden Überwachungstools wählen: – Tivoli Enterprise Monitoring Agent – CA Wily Introscope – Hyperic HQ	• „Überwachung mit dem Musterdienstprogramm "xsAdmin"" auf Seite 391 • „JMX-Sicherheit (Java Management Extensions)" auf Seite 372 • „Überwachung mit IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale" auf Seite 414 • „eXtreme Scale mit Hyperic HQ überwachen" auf Seite 424 • „eXtreme-Scale-Anwendungen mit CA Wily Introscope überwachen" auf Seite 420

Kapitel 6. Implementierungsumgebung konfigurieren

Sie können WebSphere eXtreme Scale für die Ausführung in einer eigenständigen Umgebung konfigurieren, oder Sie können eXtreme Scale für die Ausführung mit WebSphere Application Server oder WebSphere Application Server Network Deployment konfigurieren. Damit eine eXtreme-Scale-Implementierung Konfigurationsänderungen auf Server-Grid-Seite berücksichtigt, müssen Sie Prozesse erneut starten, damit diese Änderungen wirksam werden. Die Änderungen werden nicht dynamisch angewendet. Obwohl Sie auf der Clientseite die Konfigurationseinstellungen für eine vorhandene Clientinstanz nicht ändern können, können Sie jedoch unter Verwendung einer XML-Datei oder über das Programm einen neuen Client mit den erforderlichen Einstellungen erstellen. Wenn Sie einen Client erstellen, können Sie die Standardeinstellungen überschreiben, die aus der aktuellen Serverkonfiguration stammen.

Konfigurationsmethoden

XML-Dateien und Eigenschaftendateien sind die gängigsten nicht programmgesteuerten Methoden für die Konfiguration des Produkts. Im Programmierhandbuch finden Sie Informationen zu alternativen Methoden, einschließlich Anwendungs- und Systemprogrammierschnittstellen, Plug-ins und Managed Beans.

Konfiguration mit XML-Dateien

WebSphere eXtreme Scale wird mit einer Sammlung von XML-Dateien konfiguriert.

Die folgenden Konfigurationen für eXtreme Scale können mit Hilfe von XML-Dateien vorgenommen werden:

- Implementierungsrichtlinie - Zum Konfigurieren einer Implementierungsrichtlinie verwenden Sie eine XML-Deskriptordatei für Implementierungsrichtlinien. Informationen zum Definieren der Elemente und Attribute der XML-Deskriptordatei für verschiedene Anforderungen finden Sie unter "XML-Deskriptordatei für Implementierungsrichtlinien".
- ObjectGrid-Deskriptor - Zum Konfigurieren der Details einzelner ObjectGrid-Instanzen verwenden Sie eine XML-Deskriptordatei. Weitere Informationen finden Sie unter „ObjectGrid-XML-Deskriptordatei” auf Seite 145.
- Konfiguration von Entitäten - Basierend auf Ihren Anforderungen für Entitäten in der Implementierung, die in einem logischen Schema definiert sind, können Sie diese mit Hilfe annotierter Java-Klassen und/oder XML konfigurieren. Definierte Entitäten werden anschließend bei einem eXtreme-Scale-Server registriert und an BackingMaps, Indizes und andere Plug-ins gebunden. Ein Entitätsschema setzt sich aus einer Gruppe von Entitäten und den Beziehungen zwischen diesen Entitäten zusammen. Weitere Einzelheiten zum Optimieren Ihrer Entitätsschemas mit Konfigurationsoptionen finden Sie unter „Entitäten konfigurieren” auf Seite 219.
- Sicherheitskonfiguration - Sie können die Sicherheit für eine bestimmte Implementierung über XML-Konfigurationsdateien aktivieren. Weitere Informationen zu den Konfigurationsoptionen für die Sicherheitsaktivierung finden Sie unter „XML-Sicherheitsdeskriptordatei” auf Seite 379.
- Clientkonfiguration - Verwenden Sie eine Eigenschaftendatei und andere Methoden, um Clienthostnamen, Ports, Sicherheit und weitere Informationen festzule-

gen. Weitere Einzelheiten zum Anpassen von Clients mit Hilfe einer XML-Datei finden Sie unter „Clients konfigurieren” auf Seite 206.

- Konfiguration der Spring-Integration - Sie können das Spring-Framework für den Einsatz mit einer eXtreme-Scale-Implementierungsumgebung mit XML-Scripts und einer Schemadefinition sowie anderen Methoden, wie z. B. Erweiterungs-Beans und Namespace-Unterstützung, aktivieren. Weitere Einzelheiten zur Verwendung von eXtreme Scale mit dem Spring-Framework finden Sie unter „Spring-Integration konfigurieren” auf Seite 279.

Fehler in der XML-Konfiguration beheben

Gelegentlich können Sie beim Konfigurieren von eXtreme Scale ein unerwartetes Verhalten in der XML-Konfiguration beobachten.

In den folgenden Abschnitten sind verschiedene XML-Konfigurationsprobleme beschrieben, die auftreten können.

Abweichungen zwischen der XML-Datei für Implementierungsrichtlinien und der ObjectGrid-XML-Datei

Die XML-Datei für die Implementierungsrichtlinien und die ObjectGrid-XML-Datei müssen übereinstimmen. Es treten Fehler auf, wenn die ObjectGrid-Namen und Map-Namen in der XML-Datei für die Implementierungsrichtlinien und in der ObjectGrid-XML-Datei nicht identisch sind.

Ungültige BackingMap- und Map-Referenzen

Wenn die BackingMap-Liste in einer ObjectGrid-XML-Datei nicht mit der Liste der Map-Referenzen in der XML-Datei für Implementierungsrichtlinien übereinstimmt, tritt ein Fehler im Katalogserver auf.

Die folgende ObjectGrid-XML-Datei und die folgende XML-Datei für Implementierungsrichtlinien werden beispielsweise zum Starten eines Containerprozesses verwendet. Die Datei für die Implementierungsrichtlinien enthält mehr Map-Referenzen, als in der ObjectGrid-XML-Datei aufgelistet sind.

Beispiel für eine ungültige Datei "ObjectGrid.xml"

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
          <objectGrid name="accounting">
                <backingMap name="payroll" readOnly="false" />
        </objectGrid>
    </objectGrids>
</objectGridConfig>
```

Beispiel für eine ungültige Datei "deploymentPolicy.xml"

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
 xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
    <objectgridDeployment objectgridName="accounting">
        <mapSet name="mapSet1" numberOfPartitions="4" minSyncReplicas="1"
     maxSyncReplicas="2" maxAsyncReplicas="1">
            <map ref="payroll"/>
            <map ref="ledger"/>
        </mapSet>
    </objectgridDeployment>
</deploymentPolicy>
```

Nachrichten

Es wird eine Fehlernachricht in der Datei SystemOut.log aufgezeichnet, wenn die Datei für die Implementierungsrichtlinien mit der ObjectGrid-XML-Datei nicht kompatibel ist. Für das vorherige Beispiel wird die folgende Nachricht aufgezeichnet:

```
CWOBJ3179E: Die Map ledger, die im MapSet mapSet1 der Implementierungsdeskriptordatei von
ObjectGrid accounting referenziert wird, referenziert keine gültige BackingMap aus der ObjectGrid-XML.
```

Wenn in der Implementierungsrichtlinie Map-Referenzen auf BackingMaps fehlen, die in der ObjectGrid-XML-Datei aufgelistet sind, wird eine Fehlernachricht in der Datei SystemOut.log aufgezeichnet. Beispiel:

```
CWOBJ3178E: Die Map ledger in ObjectGrid accounting, die in der ObjectGrid-XML-Datei
referenziert wird, wurde nicht in der Implementierungsdeskriptordatei gefunden.
```

Problem

Die BackingMap-Liste in der ObjectGrid-XML-Datei und die Map-Referenzen in der Implementierungsrichtlinie müssen übereinstimmen.

Lösung

Stellen Sie fest, welche Liste die richtige für Ihre Umgebung ist, und korrigieren Sie die XML-Datei, die die ungültige Liste enthält.

Ungültige ObjectGrid-Namen

Der Name des ObjectGrids wird in der ObjectGrid-XML-Datei und in der XML-Dateif ür Implementierungsrichtlinien referenziert.

Nachricht

Es tritt eine Ausnahme des Typs "ObjectGridException" ein, die durch eine Ausnahme des Typs "IncompatibleDeploymentPolicyException" verursacht wird. Es folgt ein Beispiel:

Caused by: com.ibm.websphere.objectgrid.IncompatibleDeploymentPolicyException: The objectgridDeployment with objectGridName accountin does not have a corresponding objectGrid in the ObjectGrid XML.

Problem

Die ObjectGrid-XML-Datei ist die Masterliste mit ObjectGrid-Namen. Wenn eine Implementierungsrichtlinie einen ObjectGrid-Namen enthält, der nicht in der ObjectGrid-XML-Datei enthalten ist, tritt ein Fehler auf.

Lösung

Überprüfen Sie die Rechtschreibung des ObjectGrid-Namens. Entfernen Sie alle zusätzlichen Namen, bzw. fügen Sie fehlende ObjectGrid-Namen in der ObjectGrid-XML-Datei bzw. in der XML-Datei für Implementierungsrichtlinien hinzu. In der Beispielnachricht ist der ObjectGrid-Name falsch geschrieben: "accountin" anstatt "accounting".

XML-Wert eines Attributs nicht gültig

Einigen Attributen in der XML-Datei können nur bestimmte Werte zugeordnet werden. Diese Attribute haben gültige Werte, die nach Schema aufgelistet sind. Die folgende Liste enthält einige dieser Attribute:

- Attribut "authorizationMechanism" im Element "objectGrid"
- Attribut "copyMode" im Element "backingMap"
- Attribut "lockStrategy" im Element "backingMap"
- Attribut "ttlEvictorType" im Element "backingMap"
- Attribut "type" im Element "property"
- Attribut "initialState" im Element "objectGrid"
- Attribut "evictionTriggers" im Element "backingMap"

Wenn einem dieser Attribute ein ungültiger Wert zugeordnet wird, scheitert die XML-Validierung. In der folgenden XML-Beispieldatei wird der ungültige Wert INVALID_COPY_MODE verwendet:

Beispiel für INVALID_COPY_MODE

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
          <objectGrid name="accounting">
            <backingMap name="payroll" copyMode="INVALID_COPY_MODE"/>
        <objectGrid/>
   </objectGrids>
</objectGridConfig>
```

Die folgende Nachricht wird im Protokoll aufgezeichnet:

```
CWOBJ2403E: Die XML-Datei ist ungültig. Es wurde ein Fehler in
< null > in Zeile 5 gefunden. Die Fehlernachricht ist "cvc-enumeration-valid:
Value 'INVALID_COPY_MODE' is not facet-valid with respect to enumeration
'[COPY_ON_READ_AND_COMMIT, COPY_ON_READ, COPY_ON_WRITE, NO_COPY,COPY_TO_BYTES]'".
It must be a value from the enumeration.
```

Fehlende Attribute oder Tags

Wenn in einer XML-Datei die richtigen Attribute oder Tags fehlen, treten Fehler auf. In der folgenden ObjectGrid-XML-Datei fehlt beispielsweise das Abschluss-Tag </objectGrid>:

Fehlende Attribute - Beispiel-XML

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
          <objectGrid name="accounting">
            <backingMap name="payroll" />

   </objectGrids>
</objectGridConfig>
```

Die folgende Nachricht wird im Protokoll aufgezeichnet:

```
CWOBJ2403E: Die XML-Datei ist ungültig. Es wurde ein Fehler in
< null > in Zeile 7 gefunden. Die Fehlernachricht ist "The
end-tag for element type "objectGrid" must end with a '>' delimiter."
```

Es wird eine Ausnahme des Typs "ObjectGridException" für die ungültige XML-Datei ausgelöst, die den Namen der XML-Datei enthält.

Syntaxfehler

Wenn in einer XML-Datei ein ungültiges Syntaxformat verwendet wird oder Syntax fehlt, wird eine Nachricht CWOBJ2403E im Protokoll angezeigt. Die folgende Nachricht wird beispielsweise angezeigt, wenn ein Anführungszeichen in einem der XML-Attribute fehlt:

```
CWOBJ2403E: Die XML-Datei ist ungültig. Es wurde ein Fehler in < null > in
Zeile 7 gefunden. Die Fehlernachricht ist "Open quote is expected for attribute
"maxSyncReplicas" associated with an element type "mapSet"".
```

Außerdem wird eine Ausnahme des Typs "ObjectGridException" für die ungültige XML-Datei ausgelöst.

Nicht vorhandene Plug-in-Sammlung referenzieren

Wenn Sie XML für die Definition von BackingMap-Plug-ins verwenden, muss das Attribut "pluginCollectionRef" des Elements "backingMap" eine backingMapPluginCollection referenzieren. Das Attribut "pluginCollectionRef" muss mit der ID eines der backingMapPluginCollection-Elemente übereinstimmen.

Nachricht

Wenn das Attribut "pluginCollectionRef" mit keiner der IDs der backingMapPluginConfiguration-Elemente übereinstimmt, wird die folgende Nachricht oder eine ähnliche im Protokoll angezeigt:

```
[7/14/05 14:02:01:971 CDT] 686c060e XmlErrorHandl E CWOBJ9002E:
This is an English only Error message: Invalid XML file. Line: 14; URI:
null; Message: Key 'pluginCollectionRef' with
value 'bookPlugins' not found for identity constraint of
element 'objectGridConfig'.
```

Problem

Die folgende XML-Datei wird verwendet, um den Fehler zu produzieren. Beachten Sie, dass das Attribut "pluginCollectionRef" für den Namen der BackingMap "book" auf "bookPlugins" gesetzt ist und dass die einzige backingMapPluginCollection die ID "collection1" hat:

Referenzierung eines nicht vorhandenen Attributs - XML-Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
   <objectGrids>
      <objectGrid name="bookstore">
         <backingMap name="book" pluginCollectionRef="bookPlugin" />
      </objectGrid>
   </objectGrids>
   <backingMapPluginCollections>
    <backingMapPluginCollection id="collection1">
     <bean id="Evictor"
       className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" />
     </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

Lösung

Zum Beheben des Problems müssen Sie sicherstellen, dass der Wert jedes pluginCollectionRef-Attributs mit der ID eines der backingMapPluginCollection-Elemente übereinstimmt. Ändern Sie einfach den Wert des Attributs "pluginCollectionRef" in "collection1", damit dieser Fehler nicht mehr auftritt. Alternativ können Sie die ID des vorhandenen backingMapPluginCollection-Elements so ändern, dass sie dem Wert des Attributs "pluginCollectionRef" entspricht, oder ein zusätzliches Element "backingMapPluginCollection" mit einer ID hinzufügen, die dem Wert des Attributs "pluginCollectionRef" entspricht.

XML ohne Unterstützung einer Implementierung validieren

IBM Software Development Kit (SDK) Version 1.4.2 enthält Implementierungen einiger JAXP-Funktionen (Java API for XML Processing), mit denen Sie die XML anhand des Schemas validieren können.

Wenn Sie ein SDK verwenden, das diese Implementierungen nicht enthält, können die Validierungsversuche scheitern. Wenn Sie XML mit einem SDK validieren möchten, das diese Implementierungen nicht enthält, laden Sie Apache Xerces herunter, und fügen Sie die JAR-Dateien von Apache Xerces in den Klassenpfad ein.

Wenn Sie versuchen, die XML mit einem SDK zu validieren, das die erforderlichen Implementierungen nicht enthält, enthält das Protokoll den folgenden Fehler:

```
XmlConfigBuild XML validation is enabled
SystemErr R com.ibm.websphere.objectgrid
SystemErr R at com.ibm.ws.objectgrid.ObjectGridManagerImpl.getObjectGridConfigurations(ObjectGridManagerImpl.java:182)
SystemErr R at com.ibm.ws.objectgrid.ObjectGridManagerImpl.createObjectGrid(ObjectGridManagerImpl.java:309)
SystemErr R at com.ibm.ws.objectgrid.test.config.DocTest.main(DocTest.java:128)
SystemErr R Caused by: java.lang.IllegalArgumentException: No attributes are implemented
SystemErr R at org.apache.crimson.jaxp.DocumentBuilderFactoryImpl.setAttribute(DocumentBuilderFactoryImpl.java:93)
SystemErr R at com.ibm.ws.objectgrid.config.XmlConfigBuilder.<init>XmlConfigBuilder.java:133)
SystemErr R at com.ibm.websphere.objectgrid.ProcessConfigXML$2.runProcessConfigXML.java:99)...
```

Das verwendete SDK enthält nicht die Implementierung der JAXP-Funktion, die erforderlich ist, um XML-Dateien anhand des Schemas zu validieren.

Sie können dieses Problem vermeiden, indem Sie Apache Xerces herunterladen und die JAR-Dateien in den Klassenpfad einfügen. Anschließend können Sie die XML-Datei erfolgreich validieren.

Referenz der Eigenschaftendatei

Servereigenschaftendateien enthalten Einstellungen für die Ausführung der Katalogserver und Containerserver. Sie können eine Servereigenschaftendatei für eine eigenständige Konfiguration oder eine Konfiguration von WebSphere Application Server angeben. Clienteigenschaftendateien enthalten Einstellungen für Ihren Client.

Mustereigenschaftendateien

Sie können die folgenden Mustereigenschaftendateien, die im Verzeichnis *Stammverzeichnis_von_eXtreme_Scale*\properties enthalten sind, verwenden, um Ihre Eigenschaftendatei zu erstellen:

- sampleServer.properties
- sampleClient.properties

Veraltete Systemeigenschaften

-Dcom.ibm.websphere.objectgrid.CatalogServerProperties
Die Eigenschaft ist ab WebSphere eXtreme Scale Version 7.0 veraltet. Verwenden Sie stattdessen die Eigenschaft **-Dobjectgrid.server.props**.

-Dcom.ibm.websphere.objectgrid.ClientProperties
Die Eigenschaft ist ab WebSphere eXtreme Scale Version 7.0 veraltet. Verwenden Sie stattdessen die Eigenschaft **-Dobjectgrid.client.props**.

-Dobjectgrid.security.server.prop
Die Eigenschaft ist ab WebSphere eXtreme Scale Version 6.1.0.3 veraltet. Verwenden Sie stattdessen die Eigenschaft **-Dobjectgrid.server.prop**.

-serverSecurityFile
Dieses Argument ist ab WebSphere eXtreme Scale Version 6.1.0.3 veraltet. Diese Option wird an das Script `startOgServer` übergeben. Verwenden Sie stattdessen das Argument **-serverProps**.

Grids konfigurieren

Verwenden Sie eine ObjectGrid-Deskriptor-XML-Datei, um Grids, BackingMaps, Plug-ins usw. zu konfigurieren. Zum Konfigurieren von WebSphere® eXtreme Scale verwenden Sie eine ObjectGrid-Deskriptor-XML-Datei und die Anwendungsprogrammierschnittstelle (API) "ObjectGrid". Für eine verteilte Topologie benötigen Sie nicht nur eine ObjectGrid-Deskriptor-XML-Datei, sondern auch eine XML-Datei für die Implementierungsrichtlinien.

Lokale Implementierungen konfigurieren

Eine lokale, speicherinterne Konfiguration von eXtreme Scale kann über eine ObjectGrid-XML-Deskriptordatei oder über eXtreme-Scale-APIs erstellt werden.

Informationen zu diesem Vorgang

Die folgende Datei `companyGrid.xml` ist ein Beispiel für eine ObjectGrid-Deskriptor-XML. Die ersten Zeilen der Datei enthalten den erforderlichen Header für jede ObjectGrid-XML-Datei. Die Datei definiert eine ObjectGrid-Instanz mit dem Namen "CompanyGrid" und mehrere BackingMaps mit den Namen "Customer," "Item," "OrderLine" und "Order."

Datei "companyGrid.xml"

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

 <objectGrids>
<objectGrid name="CompanyGrid">
   <backingMap name="Customer" />
   <backingMap name="Item" />
   <backingMap name="OrderLine" />
   <backingMap name="Order" />
  </objectGrid>
 </objectGrids>

</objectGridConfig>
```

Übergeben Sie die XML-Datei an eine der createObjectGrid-Methoden in der Schnittstelle "ObjectGridManager. Das folgende Codemuster validiert die Datei `companyGrid.xml` anhand des XML-Schemas und erstellt die ObjectGrid-Instanz mit dem Namen "CompanyGrid." Die neu erstellte ObjectGrid-Instanz wird nicht zwischengespeichert.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid",
 new URL("file:etc/test/companyGrid.xml"), true, false);
```

Alternativ können Sie ObjectGrid-Instanzen über das Programm ohne XML erstellen. Sie können beispielsweise das folgende Code-Snippet anstelle der vorherigen XML und des vorherigen Codes verwenden.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid ("CompanyGrid", false);
BackingMap customerMap= companyGrid.defineMap("Customer");
BackingMap itemMap= companyGrid.defineMap("Item");
BackingMap orderLineMap= companyGrid.defineMap("OrderLine");
BackingMap orderMap = companyGrid.defineMap("Order");
```

Eine vollständige Beschreibung der ObjectGrid-XML-Datei finden Sie in den Referenzinformationen zur Konfiguration von eXtreme Scale.

HashIndex konfigurieren

Der HashIndex (integrierte Klasse "com.ibm.websphere.objectgrid.plugins.index-.HashIndex") ist ein MapIndexPlugin-Plug-in, das Sie der BackingMap hinzufügen können, um statische oder dynamische Indizes zu erstellen. Er unterstützt die Schnittstellen "MapIndex" und "MapRangeIndex". Durch die ordnungsgemäße Definition und Verwendung von Indizes kann die Abfrageleistung erheblich verbessert werden.

Informationen zur Indexierung finden Sie unter "Indexierung" und unter "Zusammengesetzer HashIndex". Informationen zur Verwendung der Indexierung finden Sie unter "Indexierung für den Datenzugriff ohne Schlüssel verwenden" und unter "Zusammengesetzer HashIndex".

Attribute für die Konfiguration des HashIndex

Sie können die folgenden Attribute verwenden, um den HashIndex unter Verwendung der ObjectGrid-XML-Implementierungsdeskriptordatei oder über einen programmorientierten Ansatz zu konfigurieren:

Name Gibt den Namen des Index an. Der Name muss für jede Map eindeutig sein. Der Name wird verwendet, um das Indexobjekt von der ObjectMap-Instanz für die BackingMap abzurufen.

AttributeName
Gibt die durch Kommas getrennten Namen der zu indexierenden Attribute an. Bei Feldzugriffsindizes entsprechen die Attributnamen den Feldnamen. Bei Eigenschaftszugriffsindizes sind die Attributnamen die JavaBean-kompatiblen Eigenschaftsnamen. Wenn es nur einen einzigen Attributnamen gibt, ist der HashIndex ein Einzelattributindex, und wenn dieses Attribut eine Beziehung ist, zusätzlich auch ein Beziehungsindex. Werden mehrere Attributnamen angegeben, ist der HashIndex ein zusammengesetzter Index.

FieldAccessAttribute
Wird für Maps verwendet, die keine Entitäts-Maps sind. Wenn diese Einstellung den Wert `true` hat, wird direkt über die Felder auf das Objekt zugegriffen. Wenn diese Einstellung nicht angegeben oder auf `false` gesetzt wird, wird die Getter-Methode des Attributs verwendet, um auf die Daten zuzugreifen.

POJOKeyIndex
Wird für Maps verwendet, die keine Entitäts-Maps sind. Wenn diese Einstellung auf `true` gesetzt ist, überwacht der Index selbst das Objekt im Schlüsselteil der Map. Dies ist hilfreich, wenn der Schlüssel ein zusammengesetzter Schlüssel und in den Wert kein Schlüssel eingebettet ist. Wenn

diese Einstellung nicht angegeben oder auf `false` gesetzt wird, überwacht der Index selbst das Objekt im Wertteil (value) der Map.

RangeIndex
Wenn diese Einstellung auf `true` gesetzt ist, ist die Bereichsindexierung aktiviert, und die Anwendung kann das abgerufene Indexobjekt in die Schnittstelle "MapRangeIndex" umsetzen. Wird die Eigenschaft "RangeIndex" mit dem Wert `false` konfiguriert, kann die Anwendung das abgerufene Indexobjekt nur in die Schnittstelle "MapIndex" umsetzen.

HashIndex der BackingMap hinzufügen

Es gibt verschiedene Methoden für das Hinzufügen des HashIndex zur BackingMap. Das folgende Beispiel veranschaulicht die XML-Konfigurationsmethode mit Hinzufügen statischer Index-Plug-ins:

XML-Konfigurationsmethode für das Hinzufügen des HashIndex zur BackingMap

```
<backingMapPluginCollection id="person">
   <bean id="MapIndexplugin"
         className="com.ibm.websphere.objectgrid.plugins.index.HashIndex">
         <property name="Name" type="java.lang.String" value="CODE"
                   description="index name" />
         <property name="RangeIndex" type="boolean" value="true"
                   description="true for MapRangeIndex" />
         <property name="AttributeName" type="java.lang.String" value="employeeCode"
                   description="attribute name" />
   </bean>
</backingMapPluginCollection>
```

In diesem XML-Konfigurationsbeispiel wird die integrierte Klasse "HashIndex" als Index-Plug-in verwendet. Die Klasse "HashIndex" unterstützt Eigenschaften, die die Benutzer konfigurieren können, wie z. B. Name, RangeIndex und AttributeName aus dem vorherigen Beispiel.

- Die Eigenschaft "Name" ist als `CODE` konfiguriert, einer Zeichenfolge, die dieses Index-Plug-in identifiziert. Der Wert der Eigenschaft "Name" muss innerhalb des Geltungsbereichs der BackingMap eindeutig sein und kann verwendet werden, um das Indexobjekt nach Namen von der ObjectMap-Instanz für die BackingMap abzurufen.
- Die Eigenschaft "RangeIndex" ist mit `true` konfiguriert, d. h., die Anwendung kann das abgerufene Indexobjekt in die Schnittstelle "MapRangeIndex" umsetzen. Wird die Eigenschaft "RangeIndex" mit dem Wert `false` konfiguriert, kann die Anwendung das abgerufene Indexobjekt nur in die Schnittstelle "MapIndex" umsetzen. MapRangeIndex unterstützt Funktionen, mit denen Sie Daten über Bereichsfunktionen, wie z. B. größer als und/oder kleiner als, suchen können, wohingegen die Schnittstelle "MapIndex" nur Vergleichsfunktionen unterstützt. Wenn der Index von einer Abfrage verwendet wird, muss die Eigenschaft "RangeIndex" für Einzelattributindizes mit `true` konfiguriert werden. Für einen Beziehungsindex oder einen zusammengesetzten Index muss die Eigenschaft "RangeIndex" mit `false` konfiguriert werden.
- Die Eigenschaft "AttributeName" ist mit `employeeCode` konfiguriert, d. h., das Attribut "employeeCode" des zwischengespeicherten Objekts wird verwendet, um einen Einzelattributindex zu erstellen. Wenn eine Anwendung zwischengespeicherte Objekte mit mehreren Attributen suchen muss, kann die Eigenschaft "AttributeName" auf eine durch Kommas begrenzte Liste mit Attributen gesetzt werden. Dies ergibt dann einen zusammengesetzten Index.

Zusammenfassend gesagt, das vorherige Beispiel definiert einen Bereichs-HashIndex mit einem einzigen Attribut. Es handelt sich um einen HashIndex mit einem einzigen Attribut und gleichzeitig um einen Bereichs-HashIndex.

Gegenüberstellung eines HashIndex mit einem einzigen Attribut und eines zusammengesetzten HashIndex

Wenn die Eigenschaft "AttributeName" des HashIndex mehrere Attributnamen enthält, ist der HashIndex ein zusammengesetzter Index. Enthält die Eigenschaft nur einen einzigen Attributnamen, ist der HashIndex ein Einzelattributindex. Der Wert der Eigenschaft "AttributeName" eines zusammengesetzten HashIndex kann beispielsweise `city,state,zipcode` sein. Er enthält drei Attribute, die durch Kommas voneinander getrennt sind. Wenn der Wert der Eigenschaft "AttributeName" nur aus `zipcode` besteht, hat der HashIndex nur ein einziges Attribut, d. h., er ist ein Einzelattribut-HashIndex. Das vorherige Muster ist ein HashIndex mit einem einzigen Attribut, weil die Eigenschaft "AttributeName" den Wert "employeeCode" hat, der nur einen einzigen Attributnamen einschließt.

Ein zusammengesetzter HashIndex ist eine effiziente Methode für die Suche zwischengespeicherter Objekte, wenn die Suchkriterien viele Attribute umfassen. Ein solcher Index unterstützt jedoch keine Bereichsindexierung, und seine Eigenschaft "RangeIndex" muss auf "false" gesetzt werden.

Weitere Informationen finden Sie im Abschnitt zum zusammengesetzten HashIndex im *Administratorhandbuch*.

Beziehungs-HashIndex

Wenn das indexierte Attribut eines Einzelattribut-HashIndex eine Beziehung (mit einem oder mehreren Werten) ist, ist der HashIndex ein Beziehungs-HashIndex. Für einen Beziehungs-HashIndex muss die Eigenschaft auf "false" gesetzt werden.

Ein Beziehungs-Hashindex kann Abfragen beschleunigen, die zyklische Referenzen oder die Abfragefilter IS NULL, IS EMPTY, SIZE und MEMBER OF verwenden. Weitere Informationen finden Sie in den Abschnitten zur Abfrageoptimierung mit Indizes im *Programmierhandbuch*.

Schlüssel-HashIndex

Wenn die Eigenschaft "POJOKeyIndex" von HashIndex bei Maps, die keine Entitäts-Maps sind, auf `true` gesetzt wird, ist der HashIndex ein Schlüssel-HashIndex, und der Schlüsselteil des Eintrags wird für die Indexierung verwendet. Wenn die Eigenschaft "AttributeName" von HashIndex nicht angegeben wird, wird der gesamte Schlüssel indexiert. Andernfalls kann der Schlüssel-HashIndex nur ein HashIndex mit einem einzigen Attribut sein.

Das Hinzufügen der folgenden Eigenschaft im vorherigen Muster bewirkt beispielsweise, dass aus dem HashIndex ein Schlüssel-HashIndex wird, weil die Eigenschaft "POJOKeyIndex" den Wert `true` hat.

```
<property name="POJOKeyIndex" type="boolean" value="true"
description="indicates if POJO key HashIndex" />
```

Da im vorherigen Schlüsselindexbeispiel für die Eigenschaft "AttributeName" der Wert `employeeCode` angegeben wurde, ist das Feld "employeeCode" des Schlüsselteils des Map-Eintrags das indexierte Attribut. Wenn Sie den Schlüsselindex für

den gesamen Schlüsselteil des Map-Eintrags erstellen möchten, entfernen Sie die Eigenschaft "AttributeName".

Bereichs-HashIndex

Wenn die Eigenschaft "RangeIndex" von HashIndex auf `true` gesetzt wird, ist der HashIndex ein Bereichsindex und unterstützt die Schnittstelle "MapRangeIndex". MapRangeIndex unterstützt Funktionen, mit denen Sie Daten über Bereichsfunktionen, wie z. B. größer als und/oder kleiner als, suchen können, wohingegen die Schnittstelle "MapIndex" nur Vergleichsfunktionen unterstützt. Für einen Einzelattributindex kann die Eigenschaft "RangeIndex" nur dann auf `true` gesetzt werden, wenn das indexierte Attribut den Typ "Comparable" (Vergleichbar) hat. Wird der Einzelattributindex von einer Abfrage verwendet, muss die Eigenschaft "RangeIndex" auf `true` gesetzt werden und das indexierte Attribut den Typ "Comparable" haben. Für einen Beziehungs-HashIndex und einen zusammengesetzten HashIndex muss die Eigenschaft "RangeIndex" auf `false` gesetzt werden.

Das vorherige Muster ist ein Bereichs-HashIndex, weil die Eigenschaft "RangeIndex" den Wert `true` hat.

Die folgende Tabelle enthält eine Zusammenfassung für die Verwendung eines Bereichsindex.

Tabelle 7. Unterstützung für Bereichsindizes. Unterstützung eines Bereichsindex durch HashIndex-Typen

Typ des HashIndex	**Unterstützung von Bereichsindizes**
Einzelattribut-HashIndex: indexierter Schlüssel, bzw. indexiertes Attribut hat den Typ "Comparable"	Ja
Einzelattribut-HashIndex: indexierter Schlüssel, bzw. indexiertes Attribut hat nicht den Typ "Comparable"	Nein
Zusammengesetzter HashIndex	Nein
Beziehungs-HashIndex	Nein

Abfrageoptimierung mit HashIndex

Durch die ordnungsgemäße Definition und Verwendung von Indizes kann die Abfrageleistung erheblich verbessert werden. WebSphere-eXtreme-Scale-Abfragen können integrierte HashIndex-Plug-ins verwenden, um die Leistung von Abfragen zu verbessern. Die Verwendung von Indizes kann die Abfrageleistung zwar erheblich verbessern, sich aber nachteilig auf die Leistung von Operationen für Transaktions-Maps auswirken.

Evictor konfigurieren

Evictor (Bereinigungsprogramme) können über die ObjectGrid-XML-Deskriptordatei oder über das Programm konfiguriert werden.

Informationen zu diesem Vorgang

Referenzinformationen zum Konfigurieren von Evictor mit XML finden Sie unter „ObjectGrid-XML-Deskriptordatei” auf Seite 145.

TTL-Evictor

WebSphere eXtreme Scale stellt einen Standardmechanismus für das Entfernen von Cacheeinträgen und ein Plug-in für das Erstellen angepasster Evictor (Bereinigungsprogramme) bereit. Ein Evictor steuert die Zugehörigkeit von Einträgen in jeder BackingMap-Instanz.

TTL-Evictor über das Programm aktivieren

TTL-Evictor (TimeToLive) werden BackingMap-Instanzen zugeordnet. Der Standard-Evictor verwendet eine Bereinigungsrichtlinie für jede BackingMap-Instanz, die auf der Lebensdauer (TTL, Time-to-Live) basiert. Wenn Sie einen Plug-in-fähigen Evictor bereitstellen, verwendet dieser gewöhnlich eine Bereinigungsrichtlinie, die auf der Anzahl der Einträge und nicht auf der Lebensdauer basiert.

Das folgende Code-Snippet verwendet die Schnittstelle "BackingMap", um die Verfallszeit für jeden Eintrag auf 10 Minuten nach der Eintragserstellung zu setzen.

Programmgesteuerter TTL-Evictor

```
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.TTLType;
ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid og = ogManager.createObjectGrid( "grid" );
BackingMap bm = og.defineMap( "myMap" );
bm.setTtlEvictorType( TTLType.CREATION_TIME );
bm.setTimeToLive( 600 );
```

Das Argument für die Methode "setTimeToLive" ist 600 und gibt die Lebensdauer in Sekunden an. Der vorherige Code muss vor dem Aufruf der Methode "initialize" in der ObjectGrid-Instanz aufgerufen werden. Die BackingMap-Attribute können nach der Initialisierung der ObjectGrid-Instanz nicht mehr geändert werden. Nach der Ausführung des Codes haben alle Einträge, die in die BackingMap "myMap" eingefügt werden, eine Verfallszeit. Nach Ablauf der Verfallszeit entfernt der TTL-Evictor den Eintrag.

Wenn Sie die Verfallszeit auf die letzte Zugriffszeit plus 10 Minuten setzen möchten, ändern Sie das Argument, das an die Methode "setTtlEvictorType" übergeben wird, von TTLType.CREATION_TIME in TTLType.LAST_ACCESS_TIME. Mit diesem Wert wird die Verfallszeit mit der Formel "letzte Zugriffszeit plus zehn Minuten" berechnet. Bei der Erstellung eines Eintrags entspricht die letzte Zugriffszeit der Erstellungszeit. Wenn Sie die letzte *Aktualisierung* und nicht nur den letzten *Zugriff* (unabhängig davon, ob dieser eine Aktualisierung beinhaltet) als Basis für die Verfallszeit verwenden möchten, ersetzen Sie die TTLType.LAST_ACCESS_TIME durch TTLType.LAST_UPDATE_TIME.

Wenn Sie die Einstellung TTLType.LAST_ACCESS_TIME oder TTLType.LAST_UPDATE_TIME verwenden, können Sie die Schnittstellen "ObjectMap" und "JavaMap" verwenden, um den TTL-Wert für die BackingMap zu überschreiben. Dieser Mechanismus ermöglicht einer Anwendung, für jeden erstellten Eintrag einen anderen TTL-Wert zu verwenden. Angenommen, im vorherigen Code-Snippet wurde das Attribut "ttlType" auf LAST_ACCESS_TIME und der TTL-Wert auf 10 Minuten gesetzt. In diesem Fall kann eine Anwendung den TTL-Wert für jeden Eintrag ändern, indem sie den folgenden Code vor der Erstellung bzw. Änderung eines Eintrags ausführt:

```
import com.ibm.websphere.objectgrid.Session;
import com.ibm.websphere.objectgrid.ObjectMap;
Session session = og.getSession();
ObjectMap om = session.getMap( "myMap" );
int oldTimeToLive1 = om.setTimeToLive( 1800 );
om.insert("key1", "value1" );
int oldTimeToLive2 = om.setTimeToLive( 1200 );
om.insert("key2", "value2" );
```

Im vorherigen Code-Snippet hat der Eintrag mit dem Schlüssel "key1" aufgrund des Methodenaufrufs "setTimeToLive(1800) in der ObjectMap-Instanz eine Verfallszeit, die der Einfügezeit plus 30 Minuten entspricht. Die Variable "oldTimeToLive1" wird auf 600 gesetzt, weil der TTL-Wert aus der BackingMap als Standardwert verwendet wird, wenn die Methode "setTimeToLive" in der ObjectMap-Instanz noch nicht aufgerufen wurde.

Der Eintrag mit dem Schlüssel "key2" hat aufgrund des Methodenaufrufs "setTimeToLive(1200)" in der ObjectMap-Instanz eine Verfallszeit, die der Einfügezeit plus 20 Minuten entspricht. Die Variable "oldTimeToLive2" wird auf 1800 gesetzt, weil der TTL-Wert aus dem vorherigen Aufruf der Methode "ObjectMap.setTimeToLive" den TTL-Wert auf 1800 gesetzt hat.

Das vorherige Beispiel zeigt zwei Map-Einträge, die in die Map "myMap" für die Schlüssel "key1" und "key2" eingefügt werden. Die Anwendung kann diese Map-Einträge später trotzdem aktualisieren und gleichzeitig die TTL-Werte beibehalten, die beim Einfügen jedes Map-Eintrags verwendet wurden. Das folgende Beispiel veranschaulicht, wie die TTL-Werte durch Verwendung einer in der Schnittstelle "ObjectMap" definierten Konstanten beibehalten werden können:

```
Session session = og.getSession();
ObjectMap om = session.getMap( "myMap" );
om.setTimeToLive( ObjectMap.USE_DEFAULT );
session.begin();
om.update("key1", "updated value1" );
om.update("key2", "updated value2" );
om.insert("key3", "value3" );
session.commit();
```

Da der Sonderwert ObjectMap.USE_DEFAULT im Aufruf der Methode "setTimeToLive" verwendet wird, behält der Schlüssel "key1" seinen TTL-Wert von 1800 Sekunden und der Schlüssel "key2" seinen TTL-Wert von 1200 Sekunden bei, weil diese Werte verwendet wurden, als diese Map-Einträge durch die vorherige Transaktion eingefügt wurden.

Das vorherige Beispiel zeigt auch einen neuen Map-Eintrag für den eingefügten Schlüssel "key3". In diesem Fall gibt der Sonderwert USE_DEFAULT an, dass die TTL-Standardeinstellung für diese Map zu verwenden ist. Der Standardwert wird mit dem TTL-Attribut der BackingMap definiert. Weitere Informationen zur Definition des TTL-Attributs in der BackingMap-Instanz finden Sie in der Beschreibung der Attribute der Schnittstelle "BackingMap".

Informationen zur Methode "setTimeToLive" in den Schnittstellen "ObjectMap" und "JavaMap" finden Sie in der API-Dokumentation. In der Dokumentation werden Sie darauf hingewiesen, dass eine Ausnahme des Typs "IllegalStateException" ausgelöst wird, wenn die Methode "BackingMap.getTtlEvictorType" einen anderen Wert als den Wert von TTLType.LAST_ACCESS_TIME bzw. TTLType.LAST_UPDATE_TIME zurückgibt. Die Schnittstellen "ObjectMap" und "JavaMap" können den TTL-Wert nur überschreiben, wenn die Einstellung LAST_ACCESS_TIME oder TTLType.LAST_UPDATE_TIME für den TTL-Evictor-Typ verwendet wird. Die Me-

thode "setTimeToLive" kann nicht zum Überschreiben des TTL-Werts verwendet werden, wenn die Einstellung CREATION_TIME oder NONE für den Evictor-Typ verwendet wird.

TTL-Evictor über eine XML-Konfiguration aktivieren

Anstelle der Verwendung der Schnittstelle "BackingMap" für die programmgesteuerte Definition der BackingMap-Attribute für den TTL-Evictor können Sie jede BackingMap-Instanz über eine XML-Datei konfigurieren. Der folgende Code veranschaulicht, wie Sie diese Attribute für drei verschiedene BackingMap-Maps definieren:

TTL-Evictor über XML aktivieren

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
<objectGrids>
    <objectGrid name="grid1">
        <backingMap name="map1" ttlEvictorType="NONE" />
        <backingMap name="map2" ttlEvictorType="LAST_ACCESS_TIME|LAST_UPDATE_TIME"
     timeToLive="1800" />
        <backingMap name="map3" ttlEvictorType="CREATION_TIME" timeToLive="1200" />
</objectgrid>
</objectGrids>
```

Das vorherige Beispiel zeigt, dass die BackingMap-Instanz "map1" den TTL-Evictor-Typ NONE verwendet. Die BackingMap-Instanz "map2" verwendet den TTL-Evictor-Typ LAST_ACCESS_TIME oder LAST_UPDATE_TIME (geben Sie nur eine einzige dieser Einstellungen an) und hat einen TTL-Wert von 1800 Sekunden bzw. 30 Minuten. Die BackingMap-Instanz "map3" verwendet den TTL-Evictor-Typ CREATION_TIME und hat einen TTL-Wert von 1200 Sekunden (oder 20 Minuten).

Plug-in-Evictor integrieren

Da Evictor (Bereinigungsprogramme) BackingMaps zugeordnet werden, verwenden Sie die Schnittstelle "BackingMap", um den Plug-in-Evictor anzugeben.

Optionale Plug-in-Evictor

Der TTL-Evictor verwendet eine Bereinigungsrichtlinie, die auf Zeit basiert, und die Anzahl der Einträge in der BackingMap hat keine Auswirkung auf die Verfallszeit eines Eintrags. Sie können einen optionalen Plug-in-Evictor verwenden, um Einträge auf der Basis der Anzahl vorhandener Einträge und nicht auf der Basis der Zeit zu entfernen.

Die folgenden optionalen Plug-in-Evictor stellen einige gängige Algorithmen bereit, mit denen entschieden werden kann, welche Einträge entfernt werden sollen, wenn eine BackingMap ihr Größenlimit erreicht.

- Der Evictor "LRUEvictor" verwendet einen LRU-Algorithmus (Least Recently Used), um zu entscheiden, welche Einträge entfernt werden sollen, wenn die BackingMap eine maximale Anzahl an Einträgen überschreitet.
- Der Evictor "LFUEvictor" verwendet einen LFU-Algorithmus (Least Frequently Used), um zu entscheiden, welche Einträge entfernt werden sollen, wenn die BackingMap eine maximale Anzahl an Einträgen überschreitet.

Die BackingMap informiert einen Evictor, wenn Einträge in einer Transaktion erstellt, geändert oder entfernt werden. Die BackingMap verfolgt diese Einträge und entscheidet, wann Einträge aus der BackingMap-Instanz entfernt werden müssen.

Eine BackingMap-Instanz hat keine Konfigurationsdaten für eine maximale Größe. Stattdessen werden Eigenschaften des Evictors definiert, um das Verhalten des Evictors zu steuern. Die Evictor "LRUEvictor" und "LFUEvictor" haben beide eine Eigenschaft für die maximale Größe, die den Evictor anweist, mit dem Entfernen von Einträgen zu beginnen, wenn die maximale Größe überschritten wird. Wie der TTL-Evictor entfernen auch die Evictor "LRUEvictor" und "LFUEvictor" einen Eintrag möglicherweise nicht sofort, wenn die maximale Anzahl an Einträgen erreicht ist, um die Auswirkungen auf die Leistung zu minimieren.

Wenn der LRU- bzw. LFU-Bereinigungsalgorithmus für eine bestimmte Anwendung nicht angemessen ist, können Sie eigene Evictor schreiben, um eine eigene Bereinigungsstrategie zu erstellen.

Optionale Plug-in-Evictor verwenden

Wenn Sie der BackingMap-Konfiguration einen Plug-in-Evictor hinzufügen möchten, können Sie die programmgesteuerte Konfiguration oder die XML-Konfiguration verwenden.

Plug-in-Evictor programmgesteuert integrieren

Da Evictor BackingMaps zugeordnet werden, verwenden Sie die Schnittstelle "BackingMap", um den Plug-in-Evictor anzugeben.Das folgende Code-Snippet ist ein Beispiel für die Angabe eines LRUEvictor-Evictors für die BackingMap "map1" und eines LFUEvictor-Evictors für die BackingMap-Instanz "map2":

Evictor über das Programm einfügen

```
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor;
import com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor;
ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid og = ogManager.createObjectGrid( "grid" );
BackingMap bm = og.defineMap( "map1" );
LRUEvictor evictor = new LRUEvictor();
evictor.setMaxSize(1000);
evictor.setSleepTime( 15 );
evictor.setNumberOfLRUQueues( 53 );
bm.setEvictor(evictor);
bm = og.defineMap("map2");
LFUEvictor evictor2 = new LFUEvictor();
evictor2.setMaxSize(2000);
evictor2.setSleepTime( 15 );
evictor2.setNumberOfHeaps( 211 );
bm.setEvictor(evictor2);
```

Das vorherige Snippet zeigt, dass ein LRUEvictor-Evictor für die BackingMap "map1" mit einer ungefähren maximalen Anzahl von 53.000 (53 * 1000) Einträgen verwendet wird. Der LFUEvictor-Evictor wird für die BackingMap "map2" mit einer ungefähren maximalen Anzahl von 422.000 (211 * 2000) Einträgen verwendet. Der LRU- und der LFU-Evictor haben jeweils eine Ruhezeiteigenschaft, die angibt, wie lange der Evictor ruht, bis er aktiviert wird und prüft, ob Einträge entfernt werden müssen. Die Ruhezeit wird in Sekunden angegeben. Mit einem Wert von 15 Sekunden kann erreicht werden, dass die Leistungseinbußen nicht zu hoch wer-

den und die BackingMap nicht zu groß wird. Das Ziel ist, die größtmögliche Ruhezeit zu verwenden, aber gleichzeitig zu verhindern, dass die BackingMap zu groß wird.

Die Methode "setNumberOfLRUQueues" setzt die LRUEvictor-Eigenschaft, die angibt, wie viele LRU-Warteschlangen der Evictor verwendet, um die LRU-Informationen zu verwalten. Es wird eine Sammlung von Warteschlangen verwendet, so dass nicht jeder Eintrag die LRU-Informationen in derselben Warteschlange hält. Dieser Ansatz kann die Leistung verbessern, indem die Anzahl der Map-Einträge, die in demselben Warteschlangenobjekt synchronisiert werden müssen, minimiert wird. Die Anzahl der Warteschlangen zu erhöhen, ist eine geeignete Methode, die Auswirkungen zu minimieren, die der LRU-Evictor auf die Leistung haben kann. Ein guter Ausgangspunkt für die Anzahl der Warteschlangen sind zehn Prozent der maximalen Eintragsanzahl. Die Verwendung einer Primzahl ist gewöhnlich besser als die Verwendung einer anderen Zahl. Die Methode "setMaxSize" gibt an, wie viele Einträge in jeder Warteschlange zulässig sind. Wenn eine Warteschlange die maximal zulässige Eintragsanzahl erreicht, werden die Einträge mit dem ältesten Verwendungsdatum in dieser Warteschlange entfernt, wenn der Evictor das nächste Mal prüft, ob Einträge entfernt werden müssen.

Die Methode "setNumberOfHeaps" legt mit der LFUEvictor-Eigenschaft fest, wie viele binäre Objekte im Heap-Speicher der LFUEvictor für die Verwaltung der LFU-Informationen verwendet. Auch hier wird eine Sammlung verwendet, um die Leistung zu verbessern. Ein guter Ausgangspunkt sind zehn Prozent der maximalen Eintragsanzahl, und die Verwendung einer Primzahl ist gewöhnlich besser als die Verwendung einer anderen Zahl. Die Methode "setMaxSize" gibt an, wie viele Einträge in jedem Heap-Speicher zulässig sind. Wenn ein Heap-Speicher die maximal zulässige Eintragsanzahl erreicht, werden die Einträge, die am seltensten verwendet wurden, aus diesem Heap-Speicher entfernt, wenn der Evictor das nächste Mal prüft, ob Einträge entfernt werden müssen.

XML-Konfiguration für die Integration eines Plug-in-Evictors

Anstatt verschiedene APIs für die programmgesteuerte Integration eines Evictors und die Definition seiner Eigenschaften zu verwenden, können Sie eine XML-Datei verwenden, um jede BackingMap einzeln zu konfigurieren, wie im folgenden Beispiel gezeigt wird:

Evictor über XML einfügen

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
<objectGrids>
    <objectGrid name="grid">
        <backingMap name="map1" ttlEvictorType="NONE" pluginCollectionRef="LRU" />
        <backingMap name="map2" ttlEvictorType="NONE" pluginCollectionRef="LFU" />
    </objectGrid>
</objectGrids>
<backingMapPluginCollections>
    <backingMapPlugincollection id="LRU">
        <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
            <property name="maxSize" type="int" value="1000" description="set max size for each LRU queue" />
<property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
<property name="numberOfLRUQueues" type="int" value="53" description="set number of LRU queues" />
</bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="LFU">
        <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor">
            <property name="maxSize" type="int" value="2000" description="set max size for each LFU heap" />
            <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
            <property name="numberOfHeaps" type="int" value="211" description="set number of LFU heaps" />
        </bean>
    </backingMapPluginCollection>
</backingMapPluginCollections>
</objectGridConfig>
```

Speicherbasierte Bereinigung

Alle integrierten Evictor unterstützten die speicherbasierte Bereinigung, die in der Schnittstelle "BackingMap" aktiviert werden kann, indem das Attribut "evictionTriggers" der BackingMap auf MEMORY_USAGE_THRESHOLD gesetzt wird. Weitere Informationen zum Definieren des Attributs "evictionTriggers" in BackingMap finden Sie in den Referenzinformationen zur Schnittstelle "BackingMap" und zur Konfiguration von eXtreme Scale.

Die speicherbasierte Bereinigung basiert auf einem Schwellenwert für die Auslastung des Heap-Speichers. Wenn die speicherbasierte Bereinigung in der BackingMap aktiviert ist und die BackingMap einen integrierten Evictor hat, wird der Schwellenwert für die Auslastung auf einen Standardprozentsatz des Gesamtspeichers gesetzt, wenn noch kein Schwellenwert definiert wurde.

Wenn Sie den Standardprozentsatz für den Auslastungsschwellenwert ändern möchten, setzen Sie die Eigenschaft "memoryThresholdPercentage" in den Container- und Servereigenschaftendateien für eXtreme-Scale-Serverprozesse. Zum Definieren des Schwellenwerts für die Zielauslastung in einem eXtreme-Scale-Clientprozess können Sie die MBean "MemoryPoolMXBean" verwenden. Weitere Informationen finden Sie auch in der Beschreibung der Datei "containerServer.props" und im Abschnitt zum Starten von eXtreme-Scale-Serverprozessen.

Wenn die Speicherbelegung zur Laufzeit den Schwellenwert für die Zielauslastung überschreitet, beginnen speicherbasierte Evictor mit dem Entfernen von Einträgen und versuchen, die Speicherbelegung unterhalb des Schwellenwerts für die Zielauslastung zu halten. Es gibt jedoch keine Garantien, dass die Bereinigung schnell genug ist, um potenzielle abnormale Speicherbedingungen zu verhindern, wenn die Laufzeitumgebung des Systems weiterhin so schnell Speicher belegt.

Sperrstrategie konfigurieren

Sie können eine optimistische Strategie, eine pessimistische Strategie oder eine Strategie ohne Sperren für jede BackingMap in der WebSphere eXtreme Scale-Konfiguration definieren.

Informationen zu diesem Vorgang

Sie können eine Sperrstrategie programmgesteuert oder mit XML konfigurieren. Weitere Informationen zu Sperren finden Sie in den Beschreibungen der Sperrstrategien in der *Produktübersicht*.

Vorgehensweise

- **Optimistische Sperrstrategie konfigurieren**
  - Über das Programm

    Optimistische Sperrstrategie über das Programm angeben

    ```
    import com.ibm.websphere.objectgrid.BackingMap;
    import com.ibm.websphere.objectgrid.LockStrategy;
    import com.ibm.websphere.objectgrid.ObjectGrid;
    import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
    ...
    ObjectGrid og =
     ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test");
    BackingMap bm = og.defineMap("optimisticMap");
    bm.setLockStrategy( LockStrategy.OPTIMISTIC );
    ```

  - Mit XML

Optimistische Sperrstrategie mit XML angeben

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
        <objectGrid name="test">
            <backingMap name="optimisticMap"
                lockStrategy="OPTIMISTIC"/>
        </objectGrid>
    </objectGrids>
</objectGridConfig>
```

- **Pessimistische Sperrstrategie konfigurieren**
  - Über das Programm

    Pessimistische Sperrstrategie über das Programm angeben

    ```
    import com.ibm.websphere.objectgrid.BackingMap;
    import com.ibm.websphere.objectgrid.LockStrategy;
    import com.ibm.websphere.objectgrid.ObjectGrid;
    import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
    ...
    ObjectGrid og =
     ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test");
    BackingMap bm = og.defineMap("pessimisticMap");
    bm.setLockStrategy( LockStrategy.PESSIMISTIC );
    ```

  - Mit XML

    Pessimistische Sperrstrategie mit XML angeben

    ```
    <?xml version="1.0" encoding="UTF-8"?>
    <objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
        xmlns="http://ibm.com/ws/objectgrid/config">

        <objectGrids>
            <objectGrid name="test">
                <backingMap name="pessimisticMap"
                    lockStrategy="PESSIMISTIC"/>
            </objectGrid>
        </objectGrids>
    </objectGridConfig>
    ```

- **Strategie ohne Sperren konfigurieren**
  - Über das Programm

    Strategie ohne Sperren über das Programm angeben

    ```
    import com.ibm.websphere.objectgrid.BackingMap;
    import com.ibm.websphere.objectgrid.LockStrategy;
    import com.ibm.websphere.objectgrid.ObjectGrid;
    import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
    ...
    ObjectGrid og =
     ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test");
    BackingMap bm = og.defineMap("noLockingMap");
    bm.setLockStrategy( LockStrategy.NONE );
    ```

  - Mit XML

    Strategie ohne Sperren mit XML angeben

    ```
    <?xml version="1.0" encoding="UTF-8"?>
    <objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
        xmlns="http://ibm.com/ws/objectgrid/config">

        <objectGrids>
            <objectGrid name="test">
                <backingMap name="noLockingMap"
    ```

```
                    lockStrategy="NONE"/>
            </objectGrid>
        </objectGrids>
    </objectGridConfig>
```

Nächste Schritte

Um zu vermeiden, dass eine Ausnahme des Typs "java.lang.IllegalStateException" ausgelöst wird, müssen Sie die Methode "setLockStrategy" vor den Methoden "initialize" und "getSession" für die ObjectGrid-Instanz aufrufen.

Loader konfigurieren

Die Implementierung eines Loaders (Ladeprogramm) setzt die Konfiguration verschiedener Attribute voraus.

Hinweise zum Vorherigen Laden

Loader (Ladeprogramme) sind BackingMap-Plug-ins, die aufgerufen werden, wenn Änderungen an der BackingMap vorgenommen werden oder wenn die BackingMap eine Datenanforderung nicht bedienen kann (Cachefehler). Eine Übersicht über die Interaktion von eXtreme Scale mit einem Loader finden Sie in den Informationen zu den integrierten Caching-Szenarien in der *Produktübersicht*.

Jede BackingMap hat ein boolesches Attribut "preloadMode", mit dem festgelegt werden kann, ob das vorherige Laden (Preload) einer Map asynchron durchgeführt wird oder nicht. Standardmäßig ist das "preloadMode" auf "false" gesetzt, d. h., die Initialisierung der BackingMap ist erst abgeschlossen, wenn das vorherige Laden der Map abgeschlossen ist. Die Initialisierung der BackingMap ist beispielsweise erst abgeschlossen, wenn die Methode "preloadMap" zurückkehrt. Wenn die Methode "preloadMap" sehr viele Daten aus ihrem Back-End liest und in die Map lädt, kann dieser Vorgang relativ lang dauern. In diesem Fall können Sie eine BackingMap für das asynchrone vorherige Laden der Map konfigurieren, indem Sie das Attribut "preloadMode" auf "true" setzen. Diese Einstellung bewirkt, dass der Initialisierungscode der BackingMap einen Thread startet, der die Methode "preloadMap" aufruft. Auf diese Weise kann die Initialisierung einer BackingMap abgeschlossen werden, während das vorherige Laden (Preload) der Map noch läuft.

In einem verteilten eXtreme-Scale-Szenario ist eines der Preload-Muster der Client-Preload. Im Client-Preload-Muster ist ein eXtreme-Scale-Client für den Abruf der Daten vom Back-End und das anschließende Einfügen der Daten in den verteilten eXtreme-Scale-Server unter Verwendung von DataGrid-Agenten verantwortlich. Außerdem kann ein Client-Preload in der Methode "Loader.preloadMap" in nur einer einzigen Partition ausgeführt werden. In diesem Fall wird das asynchrone Laden der Daten in das Grid sehr wichtig. Wenn der Client-Preload in demselben Thread ausgeführt wird, wird die BackingMap nie initialisiert und die Partition mit der BackingMap somit niemals online gesetzt. Deshalb kann der eXtreme-Scale-Client die Anforderung nicht an die Partition senden, was schließlich zu einer Ausnahme führt.

Wenn ein eXtreme-Scale-Client in der Methode "preloadMap" verwendet wird, müssen Sie das Attribut "preloadMode" auf "true" setzen. Alternativ können Sie einen Thread im Client-Preload-Code starten.

Das folgende Code-Snippet veranschaulicht, wie das Attribut "preloadMode" so gesetzt wird, dass das asynchrone vorherige Laden aktiviert wird:

```
BackingMap bm = og.defineMap( "map1" );
bm.setPreloadMode( true );
```

Das Attribut "preloadMode" kann auch über eine XML-Datei gesetzt werden, wie im folgenden Beispiel demonstriert wird:

```
<backingMap name="map1" preloadMode="true" pluginCollectionRef="map1"
 lockStrategy="OPTIMISTIC" />
```

TxID und die Verwendung der Schnittstelle "TransactionCallback"

An die Methoden "get" und "batchUpdate" in der Schnittstelle "Loader" wird ein TxID-Objekt übergeben, das die Session-Transaktion darstellt, die die Ausführung der Operation "get" bzw. "batchUpdate" erfordert. Die Methoden "get" und "batchUpdate" können pro Transaktion mehrfach aufgerufen werden. Deshalb werden transaktionsbezogene Objekte, die der Loader benötigt, gewöhnlich in einem Slot des TxID-Objekts verwaltet. Ein JDBC-Loader (Java Database Connectivity) wird verwendet, um zu veranschaulichen, wie ein Loader die Schnittstellen "TxID" und "TransactionCallback" verwendet.

Es ist auch möglich, mehrere ObjectGrid-Maps in derselben Datenbank zu speichern. Jede Map hat einen eigenen Loader, und jeder Loader muss möglicherweise eine Verbindung zu derselben Datenbank herstellen. Wenn Verbindungen zu derselben Datenbank hergestellt werden, möchte jeder Loader dieselbe JDBC-Verbindung verwenden, so dass die Änderungen an den einzelnen Tabellen im Rahmen derselben Datenbanktransaktion festgeschrieben werden. Gewöhnlich schreibt die Person, die die Loader-Implementierung schreibt, auch die TransactionCallback-Implementierung. Bei der Erweiterung der Schnittstelle "TransactionCallback" empfiehlt es sich, Methoden hinzuzufügen, die der Loader benötigt, um eine Datenbankverbindung anzufordern und vorbereitete Anweisungen zwischenzuspeichern. Der Grund für diese Vorgehensweise wird deutlich, wenn Sie sehen, wie die Schnittstellen "TransactionCallback" und "TxID" vom Loader verwendet werden.

Beispiel: Der Loader erfordert eine Erweiterung der Schnittstelle "TransactionCallback" wie die folgende:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import com.ibm.websphere.objectgrid.TxID;
public interface MyTransactionCallback extends TransactionCallback
{
    Connection getAutoCommitConnection(TxID tx, String databaseName) throws SQLException;
    Connection getConnection(TxID tx, String databaseName, int isolationLevel ) throws SQLException;
    PreparedStatement getPreparedStatement(TxID tx, Connection conn, String tableName, String sql) throws SQLException;
Collection getPreparedStatementCollection( TxID tx, Connection conn,  String tableName );
}
```

Mit Hilfe dieser neuen Methoden können die Methoden "get" und "batchUpdate" des Loaders wie folgt eine Verbindung anfordern:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import com.ibm.websphere.objectgrid.TxID;
private Connection getConnection(TxID tx, int isolationLevel)
{
    Connection conn = ivTcb.getConnection(tx, databaseName, isolationLevel );
    return conn;
}
```

Im vorherigen Beispiel und den Beispielen, die noch folgen, sind "ivTcb" und "ivOcb" Instanzvariablen des Loaders, die gemäß der Beschreibung im Abschnitt

"Hinweise zum vorherigen Laden" beschrieben wurden. Die Variable "ivTcb" ist eine Referenz auf die Schnittstelle "MyTransactionCallback" und die Variable "ivOcb"eine Referenz auf die MyOptimisticCallback-Instanz. Die Variable "databaseName" ist eine Instanzvariable des Loaders, die als Loader-Eigenschaft während der Initialisierung der BackingMap gesetzt wurde. Das Argument "isolationLevel" ist eine der Konstanten der JDBC-Verbindung, die für die verschiedenen von JDBC unterstützten Isolationsstufen definiert sind. Wenn der Loader eine optimistische Implementierung nutzt, verwendet die Methode "get" gewöhnlich eine JDBC-Verbindung mit automatischem Festschreiben, um die Daten aus der Datenbank abzurufen. In diesem Fall kann der Loader eine Methode "getAutoCommitConnection" haben, die wie folgt implementiert ist:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import com.ibm.websphere.objectgrid.TxID;
private Connection getAutoCommitConnection(TxID tx)
{
    Connection conn = ivTcb.getAutoCommitConnection(tx, databaseName);
    return conn;
}
```

Rufen Sie die Methode "batchUpdate" wieder auf, die die folgende switch-Anweisung enthält:

```
switch ( logElement.getType().getCode() )
{
    case LogElement.CODE_INSERT:
        buildBatchSQLInsert( tx, key, value, conn );
        break;
    case LogElement.CODE_UPDATE:
        buildBatchSQLUpdate( tx, key, value, conn );
        break;
    case LogElement.CODE_DELETE:
        buildBatchSQLDelete( tx, key, conn );
        break;
}
```

Jede der buildBatchSQL-Methoden verwendet die Schnittstelle "MyTransactionCallback", um eine vorbereitete Anweisung abzurufen. Im Folgenden sehen Sie ein Code-Snippet, das die Methode "buildBatchSQLUpdate" zeigt, die eine SQL-Anweisung "update" für die Aktualisierung eines EmployeeRecord-Eintrags und das Hinzufügen dieses Eintrags für die Aktualisierung im Stapelbetrieb erstellt:

```
private void buildBatchSQLUpdate( TxID tx, Object key, Object value, Connection conn )
throws SQLException, LoaderException
{
    String sql = "update EMPLOYEE set LASTNAME = ?, FIRSTNAME = ?, DEPTNO = ?,
    SEQNO = ?, MGRNO = ? where EMPNO = ?";
    PreparedStatement sqlUpdate = ivTcb.getPreparedStatement( tx, conn,
   "employee", sql );
    EmployeeRecord emp = (EmployeeRecord) value;
    sqlUpdate.setString(1, emp.getLastName());
    sqlUpdate.setString(2, emp.getFirstName());
    sqlUpdate.setString(3, emp.getDepartmentName());
    sqlUpdate.setLong(4, emp.getSequenceNumber());
    sqlUpdate.setInt(5, emp.getManagerNumber());
    sqlUpdate.setInt(6, key);
    sqlUpdate.addBatch();
}
```

Nachdem die batchUpdate-Schleife alle vorbereiteten Anweisungen erstellt hat, ruft sie die Methode "getPreparedStatementCollection" auf. Diese Methode ist wie folgt implementiert:

```
private Collection getPreparedStatementCollection( TxID tx, Connection conn )
{
    return ( ivTcb.getPreparedStatementCollection( tx, conn, "employee" ) );
}
```

Wenn die Anwendung die Methode "commit" in Session aufruft, ruft der Session-Code die Methode "commit" in der Methode "TransactionCallback" auf, nachdem sie alle von der Transaktion vorgenommenen Änderungen mit Push an den Loader für jede Map übertragen hat, die von der Transaktion geändert wurde. Da alle Loader die Methode "MyTransactionCallback" verwenden, um die erforderlichen Verbindungen und vorbereiteten Anweisungen abzurufen, weiß die Methode "TransactionCallback", welche Verbindung verwendet werden muss, um die Festschreibung der Änderungen im Back-End anzufordern. Die Erweiterung der Schnittstelle "TransactionCallback" mit den einzelnen von den Loadern benötigten Methoden hat somit die folgenden Vorteile:

- Das TransactionCallback-Objekt kapselt die Verwendung von TxID-Slots für transaktionsbezogene Daten, und der Loader erfordert keine Informationen zu den TxID-Slots. Der Loader muss lediglich die Methoden kennen, die TransactionCallback über die Schnittstelle "MyTransactionCallback" für die vom Loader benötigten unterstützenden Funktionen hinzugefügt werden.
- Das TransactionCallback-Objekt kann sicherstellen, dass die gemeinsame Verbindungsnutzung für alle Loader, die Verbindungen zu demselben Back-End herstellen, stattfindet, so dass ein zweiphasiges Festschreibungsprotokoll umgangen werden kann.
- Das TransactionCallback-Objekt kann über eine Commit- oder Rollback-Operation, die bei Bedarf in der Verbindung aufgerufen wird, sicherstellen, dass die Verbindungsherstellung zum Back-End abgeschlossen wird.
- TransactionCallback stellt sicher, dass eine Bereinigung der Datenbankressourcen stattfindet, wenn eine Transaktion abgeschlossen wird.
- TransactionCallback verheimlicht, wenn eine verwaltete Verbindung von einer verwalteten Umgebung wie WebSphere Application Server oder einem anderen J2EE-kompatiblen (Java 2 Platform, Enterprise Edition) Anwendungsserver abgerufen wird. Auf diese Weise kann derselbe Loader-Code in verwalteten und nicht verwalteten Umgebungen verwendet werden. Nur das TransactionCallback-Plug-in muss geändert werden.
- Ausführliche Informationen zur Verwendung der TxID-Slots für transaktionsbezogene Daten durch die TransactionCallback-Implementierung finden Sie in der Beschreibung des TransactionCallback-Plug-ins.

OptimisticCallback

Wie bereits erwähnt, kann der Loader einen optimistischen Ansatz für die Steuerung des gemeinsamen Zugriffs verwenden. In diesem Fall muss das Beispiel für die Methode "buildBatchSQLUpdate" geringfügig geändert werden, um einen optimistischen Ansatz zu implementieren. Es gibt verschiedene Methoden für die Verwendung eines optimistischen Ansatzes. Eine typische Methode ist die Verwendung einer Zeitmarken- oder Folgenummernspalte für die Versionssteuerung jeder Aktualisierung der Spalte. Angenommen, die Tabelle "employee" hat eine Folgenummernspalte, deren Wert jedesmal um eins erhöht wird, wenn die Zeile aktualisiert wird. In diesem Fall können Sie die Signatur der Methode "buildBatchSQLUpdate" so ändern, dass das LogElement-Objekt an Stelle des Schlüssel/Wert-Paars an sie übergeben wird. Außerdem muss sie das OptimisticCallback-Objekt verwenden, das in die BackingMap integriert wird, um das Anfangsversionsobjekt abzurufen und das Versionsobjekt zu aktualisieren. Im Folgenden sehen Sie ein Beispiel

für eine geänderte Methode "buildBatchSQLUpdate", die die Instanzvariable "ivOcb" verwendet, die gemäß der Beschreibung im Abschnitt zu preloadMap initialisiert wurde:

Beispiel für den geänderten Code der Methode batch-update

```
private void buildBatchSQLUpdate( TxID tx, LogElement le, Connection conn )
 throws SQLException, LoaderException
{
    // Anfangsversionsobjekt abrufen, wenn dieser Map-Eintrag
    // in der Datenbank gelesen oder aktualisiert wurde.
Employee emp = (Employee) le.getCurrentValue();
    long initialVersion = ((Long) le.getVersionedValue()).longValue();
    // Versionsobjekt aus dem aktualisierten Employee-Objekt für die
    // SQL-Operation "update" abrufen.
Long currentVersion = (Long)ivOcb.getVersionedObjectForValue( emp );
    long nextVersion = currentVersion.longValue();
    // Jetzt die SQL-Operation "update" erstellen, die das Versionsobjekt
    // in der WHERE-Klausel für optimistische Prüfung enthält.
String sql = "update EMPLOYEE set LASTNAME = ?, FIRSTNAME = ?,
    DEPTNO = ?,SEQNO = ?, MGRNO = ? where EMPNO = ? and SEQNO = ?";
    PreparedStatement sqlUpdate = ivTcb.getPreparedStatement( tx, conn,
   "employee", sql );
    sqlUpdate.setString(1, emp.getLastName());
    sqlUpdate.setString(2, emp.getFirstName());
    sqlUpdate.setString(3, emp.getDepartmentName());
    sqlUpdate.setLong(4, nextVersion );
    sqlUpdate.setInt(5, emp.getManagerNumber());
    sqlUpdate.setInt(6, key);
    sqlUpdate.setLong(7, initialVersion);
    sqlUpdate.addBatch();
}
```

Das Beispiel zeigt, dass das LogElement-Objekt verwendet wird, um den Anfangsversionswert abzurufen. Wenn die Transaktion zum ersten Mal auf den Map-Eintrag zugreift, wird ein LogElement-Objekt mit dem Anfangs-Employee-Objekt erstellt, das aus der Map abgerufen wurde. Außerdem wird das Anfangs-Employee-Objekt an die Methode "getVersionedObjectForValue" in der Schnittstelle "OptimisticCallback" übergeben und das Ergebnis im LogElement-Objekt gespeichert. Diese Verarbeitung findet statt, bevor eine Anwendung eine Referenz auf das Anfangs-Employee-Objekt erhält und die Chance hat, eine Methode aufzurufen, die den Status des Anfangs-Employee-Objekts zu ändern.

Das Beispiel zeigt, dass der Loader die Methode "getVersiondObjectForValue" verwendet, um das Versionsobjekt für das aktuelle aktualisierte Employee-Objekt abzurufen. Vor dem Aufruf der Methode "batchUpdate" in der Schnittstelle "Loader" ruft eXtreme Scale die Methode "updateVersionedObjectForValue" in der Schnittstelle "OptimisticCallback" auf, um die Generierung eines neuen Versionsobjekts für das aktualisierte Employee-Objekt anzufordern. Wenn die Methode "batchUpdate" zu ObjectGrid zurückkehrt, wird das LogElement-Objekt mit dem aktuellen Versionsobjekt aktualisiert und als neues Anfangsversionsobjekt festgelegt. Dieser Schritt ist erforderlich, weil die Anwendung die Methode "flush" für die Map an Stelle der Methode "commit" für die Session aufgerufen haben kann. Der Loader kann von einer einzelnen Transaktion mehrfach für denselben Schlüssel aufgerufen werden. Aus diesem Grund stellt eXtreme Scale sicher, dass das LogElement-Objekt jedesmal, wenn die Zeile in der Tabelle "employee" aktualisiert wird, mit dem neuen Versionsobjekt aktualisiert wird.

Jetzt hat der Loader das Anfangsversionsobjekt und das Folgeversionsobjekt und kann eine SQL-Anweisung "SQL" ausführen, die die Spalte SEQNO auf den Wert des Folgeversionsobjekts setzt und den Wert des Anfangsversionsobjekts in der WHERE-Klausel verwendet. Dieser Ansatz wird manchmal auch als überqualifi-

zierte update-Anweisung bezeichnet. Die Verwendung der überqualifizierten update-Anweisung ermöglicht der relationalen Datenbank sicherzustellen, dass die Zeile in der Zeit zwischen dem Lesen der Daten aus der Datenbank und dem Aktualisieren der Datenbank durch die Transaktion nicht geändert wurde. Wenn die Zeile von einer anderen Transaktion geändert wurde, gibt Zählerbereich, der von der Aktualisierung im Stapelbetrieb zurückgegeben wird, an, dass keine Zeilen für diesen Schlüssel geändert wurden. Der Loader muss sicherstellen, dass die SQL-Operation "update" die Zeile geändert hat. Ist dies nicht der Fall, zeigt der Loader eine Ausnahme vom Typ "com.ibm.websphere.objectgrid.plugins.OptimisticCollisionException" an, um das Session-Objekt darüber zu informieren, dass die Methode "batchUpdate" fehlgeschlagen ist, weil mehrere gleichzeitig ausgeführte Transaktionen versuchen, dieselbe Zeile in der Datenbanktabelle zu aktualisieren. Diese Ausnahme bewirkt, dass eine Rollback-Operation für das Session-Objekt durchgeführt wird, und die Anwendung muss die vollständige Transaktion wiederholen. Die Begründung ist, dass die Wiederholung erfolgreich ist, und deshalb wird dieser Ansatz auch als optimistischer Ansatz bezeichnet. Der optimistische Ansatz bietet eine bessere Leistung, wenn die Daten nur selten geändert werden und nur selten gleichzeitig ausgeführte Transaktionen versuchen, dieselbe Zeile zu aktualisieren.

Es ist wichtig, dass der Loader mit dem Parameter "key" des Konstruktors "OptimisticCollisionException" angibt, welcher Schlüssel bzw. welche Gruppe von Schlüsseln für das Fehlschlagen der optimistischen batchUpdate-Methode verantwortlich ist. Der Parameter "key" kann das Schlüsselobjekt selbst oder ein Bereich von Schlüsselobjekten sein, falls mehrere Schlüssel zum Fehlschlagen der optimistischen Aktualisierung geführt haben. eXtreme Scale verwendet die Methode "getKey" des Konstruktors "OptimisticCollisionException", um festzustellen, welche Map-Einträge veraltete Daten enthalten und deshalb zur Ausnahme geführt haben. Ein Teil der Rollback-Verarbeitung besteht darin, dass alle veralteten Map-Einträge aus der Map entfernt werden. Das Entfernen veralteter Einträge ist erforderlich, damit alle nachfolgenden Transaktionen, die auf dieselben Schlüssel zugreifen, bewirken, dass die Methode "get" der Schnittstelle "Loader" aufgerufen wird, um die Map-Einträge mit den aktuellen Daten aus der Datenbank zu aktualisieren.

Im Folgenden sind weitere Möglichkeiten beschrieben, mit denen ein Loader einen optimistischen Ansatz implementieren kann:

- Es ist keine Zeitmarken- oder Folgenummernspalte vorhanden. In diesem Fall gibt die Methode "getVersionObjectForValue" in der Schnittstelle "OptimisticCallback" einfach das Wertobjekt selbst als Version zurück. Bei diesem Ansatz muss der Loader eine WHERE-Klausel erstellen, die alle Felder des Anfangsversionsobjekts enthält. Dieser Ansatz ist nicht effizient, und nicht alle Spaltentypen eignen sich für die Verwendung in der WHERE-Klausel einer überqualifizierten SQL-Anweisung "update". Deshalb wird dieser Ansatz gewöhnlich nicht verwendet.
- Es ist keine Zeitmarken- oder Folgenummernspalte vorhanden. Anders als beim vorherigen Ansatz enthält die WHERE-Klausel jedoch die Wertfelder, die von der Transaktion geändert wurden. Eine Methode zur Erkennung der geänderten Felder ist das Einstellen des Kopiermodus in der BackingMap auf "CopyMode.COPY_ON_WRITE". Dieser Kopiermodus erfordert, dass eine Value-Schnittstelle an die Methode "setCopyMode" in der Schnittstelle "BackingMap" übergeben wird. Die BackingMap erstellt dynamische Proxy-Objekte, die die angegebene Value-Schnittstelle implementieren. Mit diesem Kopiermodus kann der Loader jeden Wert in ein com.ibm.websphere.objectgrid.plugins.ValueProxyInfo-Objekt umsetzen. Die Schnittstelle "ValueProxyInfo" hat eine Methode, die dem Loader ermöglicht, die Liste der Attributnamen abzurufen, die von der Transaktion geändert wurden. Mit dieser Methode kann der Loader die get-Methoden in der

Value-Schnittstelle für die Attributnamen aufrufen, um die geänderten Daten abzurufen und eine SQL-Anweisung "update" zu erstellen, die nur die geänderten Attribute setzt. Die WHERE-Klausel kann jetzt so erstellt werden, dass sie die Primärschlüsselspalte und alle geänderten Attributspalten enthält. Dieser Ansatz ist effizienter als der vorherige Ansatz, erfordert aber, dass mehr Code im Loader geschrieben werden muss, und kann einen größeren Cache für vorbereitete Anweisungen erfordern, damit die verschiedenen Permutationen behandelt werden können. Diese Einschränkung sollte jedoch kein Problem darstellen, wenn Transaktionen gewöhnlich nur wenige Attribute ändern.

- Einige relationale Datenbanken haben eine API für die Unterstützung der automatischen Verwaltung von Spaltendaten, die für eine optimistische Versionssteuerung hilfreich ist. Lesen Sie in der Dokumentation zu Ihrer Datenbank nach, ob diese Möglichkeit existiert.

Unterstützung für Write-behind-Loader konfigurieren

Sie können die Write-behind-Unterstützung über die ObjectGrid-XML-Deskriptordatei oder programmgesteuert über die Schnittstelle "BackingMap" aktivieren.

Verwenden Sie die ObjectGrid-XML-Deskriptordatei oder den programmgesteuerten Ansatz über die Schnittstelle "BackingMap", um die Write-behind-Unterstützung zu aktivieren.

ObjectGrid-XML-Deskriptordatei

Wenn Sie ein ObjectGrid über eine ObjectGrid-XML-Deskriptordatei konfigurieren, wird der Write-behind-Loader über das Attribut "writeBehind" im Tag "backingMap" aktiviert. Es folgt ein Beispiel:

```
<objectGrid name="library" >
     <backingMap name="book" writeBehind="T300;C900" pluginCollectionRef="bookPlugins"/>
```

Im vorherigen Beispiel wird die Write-behind-Unterstützung für die BackingMap "book" mit dem Parameter "T300;C900" aktiviert. Das Attribut "writeBehind" gibt die maximal zulässige Aktualisierungszeit und/oder eine maximale Anzahl an Schlüsselaktualisierungen an. Der Parameter "Write-behind" hat das folgende Format:

```
::= <Standardwerte> | <Aktualisierungszeit> | <Schlüsselaktualisierungsanzahl> | <Aktualisierungszeit> ";"
<Schlüsselaktualisierungsanzahl>::= "T" <positive ganze Zahl>::= "C" <positive ganze Zahl>::= ""
```

- Attribut "writeBehind"
- Aktualisierungszeit
- Schlüsselaktualisierungsanzahl
- Standardwerte

Aktualisierungen im Loader finden statt, wenn eines der folgenden Ereignisse eintritt:

1. Die maximale Aktualisierungszeit in Sekunden seit der letzten Aktualisierung ist abgelaufen.
2. Die Anzahl aktualisierter Schlüssel in der Warteschlangen-Map hat die maximal zulässige Anzahl an Schlüsselaktualisierungen erreicht.

Diese Parameter sind lediglich Hinweise. Der echte Aktualisierungszähler und die echte Aktualisierungszeit liegen nah bei den Parametern. Es ist jedoch nicht garantiert, dass der echte Aktualisierungszähler und die echte Aktualisierungszeit den definierten Parametern entsprechen. Außerdem könnte die erste Write-behind-Aktualisierung erst nach zwei Aktualisierungszeitintervallen stattfinden. Dies ist dar-

auf zurückzuführen, dass ObjectGrid die Startzeit für die Aktualisierung zufällig wählt, so dass nicht alle Partitionen gleichzeitig auf die Datenbank zugreifen.

Im vorherigen Beispiel (T300;C900) schreibt der Loader die Daten 300 Sekunden nach der letzten Aktualisierung bzw. bei 900 zu aktualisierenden Schlüsseln in die Datenbank. Die Standardaktualisierungszeit sind 300 Sekunden, und die Standardanzahl der Schlüsselaktualisierungen ist 1000.

Write-behind-Caching

Sie können Write-behind-Caching verwenden, um die Kosten für die Aktualisierung einer Datenbank, die Sie als Back-End verwenden, zu reduzieren.

Einführung

Beim Write-behind-Caching werden Aktualisierungen für das Loader-Plug-in asynchron in die Warteschlange eingereiht. Sie können die Leistung von Aktualisierungs-, Einfüge- und Entfernungsoperationen für die Map verbessern, indem Sie die eXtreme-Scale-Transaktion von der Datenbanktransaktion entkoppeln. Die asynchrone Aktualisierung wird nach einer zeitbasierten Verzögerung (z. B. fünf Minuten) oder einer eintragsbasierten Verzögerung (z. B. 1000 Einträge) durchgeführt.

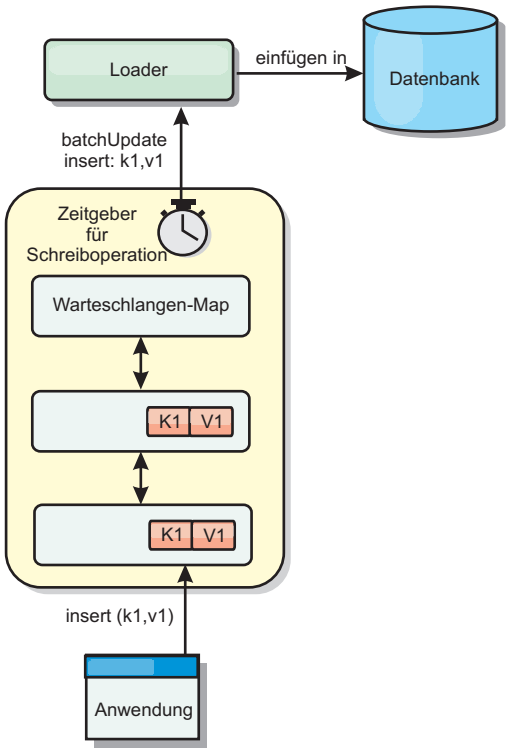


Abbildung 7. Write-behind-Caching

Bei der Write-behind-Konfiguration in einer BackingMap wird ein Thread zwischen dem Loader (Ladeprogramm) und der Map erstellt. Anschließend delegiert der Loader Datenanforderungen über den Thread gemäß den Konfigurationseinstellungen in der Methode "BackingMap.setWriteBehind". Wenn eine eXtreme-Scale-Transaktion einen Eintrag in einer Map einfügt, aktualisiert oder entfernt, wird ein LogElement-Objekt für jeden dieser Datensätze erstellt. Diese Elemente werden an den Write-behind-Loader gesendet und in eine spezielle ObjectMap, eine so genannte Warteschlangen-Map, eingereiht. Jede BackingMap mit aktivierter Write-

behind-Einstellung hat ihre eigenen Warteschlangen-Maps. Ein Write-behind-Thread entfernt die in die Warteschlange eingereihten Daten aus den Warteschlangen-Maps und überträgt Sie mit Push in den echten Back-End-Loader.

Der Write-behind-Loader sendet nur LogElement-Objekte der Typen "insert" (Einfügen), "update" (Aktualisieren) und "delete" (Löschen) an den echten Loader. Alle anderen Typen von LogElement-Objekten, wie z. B. EVICT, werden ignoriert.

Vorteile

Das Aktivieren der Write-behind-Unterstützung hat die folgenden Vorteile:

- **Isolation von Back-End-Fehlern:** Durch das Write-behind-Caching können Back-End-Fehler isoliert werden. Wenn die Back-End-Datenbank ausfällt, werden Aktualisierungen in die Warteschlangen-Map eingereiht. Die Anwendungen können weiterhin Transaktionen an eXtreme Scale senden. Nach der Wiederherstellung des Back-Ends werden die Daten in der Warteschlangen-Map mit Push an das Back-End übertragen.
- **Geringere Back-End-Last:** Der Write-behind-Loader fasst die Aktualisierungen auf Schlüsselbasis so zusammen, dass nur eine einzige zusammenfasste Aktualisierung pro Schlüssel in der Warteschlangen-Map vorhanden ist. Bei dieser Zusammenfassung verringert sich die Anzahl der Aktualisierungen für die Back-End-Datenbank.
- **Verbesserte Transaktionsleistung:** Die Zeiten einzelner eXtreme-Scale-Transaktionen verringern sich, weil sie nicht auf die Synchronisation der Daten mit dem Back-End warten müssen.

Hinweise zum Anwendungsdesign

Das Aktivieren der Write-behind-Unterstützung ist zwar einfach, aber eine Anwendung mit Write-behind-Unterstützung zu entwerfen, bedarf sorgfältiger Überlegungen. Ohne Write-behind-Unterstützung ist die Back-End-Transaktion in die ObjectGrid-Transaktion eingeschlossen. Die ObjectGrid-Transaktion wird vor der Back-End-Transaktion gestartet und endet erst nach Abschluss der Back-End-Transaktion.

Wenn die Write-behind-Unterstützung aktiviert ist, endet die ObjectGrid-Transaktion vor dem Start der Back-End-Transaktion. Die ObjectGrid-Transaktion und die Back-End-Transaktion sind entkoppelt.

Referenzielle Integritätsbedingungen

Jede BackingMap, die mit Write-behind-Unterstützung konfiguriert ist, hat einen eigenen Write-behind-Thread, der die Daten mit Push an das Back-End überträgt. Deshalb werden die Daten, die in einer einzigen ObjectGrid-Transaktion in verschiedenen Maps aktualisiert wurden, im Back-End in verschiedenen Back-End-Transaktionen aktualisiert. Beispiel: Transaktion T1 aktualisiert den Schlüssel "key1" in der Map "Map1" und den Schlüssel "key2" in der Map "Map2". Die Aktualisierungen von Schlüssel key1 in der Map Map1 und von Schlüssel "key2" in der Map "Map2" werden in einer jeweils anderen Back-End-Transaktion von einem jeweils anderen Write-behind-Thread durchgeführt. Wenn es Beziehungen zwischen den in Map1 und Map2 gespeicherten Daten, wie z. B. Integritätsbedingungen über Fremdschlüssel, im Back-End gibt, können die Aktualisierungen fehlschlagen.

Beim Design der referenziellen Integritätsbedingungen in Ihrer Back-End-Datenbank müssen Sie sicherstellen, dass solche nicht ausführbaren Aktualisierungen zugelassen werden.

Sperrverhalten von Warteschlangen-Maps

Ein weiterer wichtiger Unterschied im Transaktionsverhalten ist das Sperrverhalten. ObjectGrid unterstützt drei verschiedene Sperrstrategien: PESSIMISTIC (Pessimistisch), OPTIMISITIC (Optimistisch) und NONE (Keine). Die Write-behind-Warteschlangen-Map verwendet die pessimistische Sperrstrategie, unabhängig davon, welche Sperrstrategie für die zugehörige BackingMap konfiguriert ist. Es gibt zwei verschiedene Typen von Operationen, die eine Sperre für die Warteschlangen-Map anfordern:

- Wenn eine ObjectGrid-Transaktion festgeschrieben wird oder eine Flush-Operation (Map-Flush oder Sitzungs-Flush) stattfindet, liest die Transaktion den Schlüssel in der Warteschlangen-Map und setzt eine S-Sperre für den Schlüssel.
- Wenn eine ObjectGrid-Transaktion festgeschrieben wird, versucht die Transaktion die S-Sperre für den Schlüssel in eine X-Sperre zu aktualisieren.

Anhand dieses zusätzlichen Verhaltens für die Warteschlangen-Map sind einige Unterschiede im Sperrverhalten erkennbar.

- Wenn die Benutzer-Map mit einer pessimistischen Sperrstrategie konfiguriert ist, sind die Unterschiede im Sperrverhalten nicht gravierend. Bei jedem Aufruf einer Flush- oder Festschreiboperation (Commit) wird eine S-Sperre für denselben Schlüssel in der Warteschlangen-Map gesetzt. Während der Festschreibung wird nicht nur eine X-Sperre für den Schlüssel in der Benutzer-Map, sondern auch für den Schlüssel in der Warteschlangen-Map angefordert.
- Wenn die Benutzer-Map mit einer optimistischen Sperrstrategie oder ohne Sperrstrategie konfiguriert ist, folgt die Benutzertransaktion dem Muster der pessimistischen Sperrstrategie. Bei jedem Aufruf einer Flush- oder Festschreiboperation (Commit) wird eine S-Sperre für denselben Schlüssel in der Warteschlangen-Map angefordert. Während der Festschreibung wird in derselben Transaktion eine X-Sperre für den Schlüssel in der Warteschlangen-Map angefordert.

Transaktionswiederholungen im Loader

ObjectGrid unterstützt keine zweiphasigen Transaktionen und keine XA-Transaktionen. Der Write-behind-Thread entfernt Datensätze aus der Warteschlangen-Map und aktualisiert die Datensätze im Back-End. Wenn der Server mitten in der Transaktion ausfällt, können einige Back-End-Aktualisierungen verloren gehen.

Der Write-behind-Loader versucht automatisch, fehlgeschlagene Transaktionen erneut zu schreiben, und sendet eine unbestätigte Protokollfolge an das Back-End, um einen Datenverlust zu verhindern. Diese Aktion erfordert, dass der Loader idempotent ist, d. h., wenn Loader.batchUpdate(TxId, LogSequence) zweimal mit demselben Wert aufgerufen wird, liefen diese Aufrufe dasselbe Ergebnis wie ein einmaliger Aufruf. Loader-Implementierungen müssen zum Aktivieren dieses Features die Schnittstelle "RetryableLoader" implementieren. Weitere Einzelheiten finden Sie in der API-Dokumentation.

Ausfall des Loaders

Das Loader-Plug-in kann ausfallen, wenn es nicht mit dem Datenbank-Back-End kommunizieren kann. Dies kann passieren, wenn der Datenbankserver oder die Netzverbindung inaktiv ist. Der Write-behind-Loader reiht die Aktualisierungen in

eine Warteschlange ein und versucht anschließend in regelmäßigen Abständen, die Datenänderungen mit Push an den Loader zu übertragen. Der Loader muss die ObjectGrid-Laufzeitumgebung darüber benachrichtigen, dass ein Problem mit der Datenbankkonnektivität vorliegt, indem es eine Ausnahme vom Typ "LoaderNotAvailableException" auslöst.

Deshalb muss die Loader-Implementierung in der Lage sein, einen Datenfehler von einem physischen Ausfall des Loaders zu unterscheiden. Bei Datenfehlern muss eine Ausnahme des Typs "LoaderException" oder "OptimisticCollisionException" ausgelöst bzw. erneut ausgelöst werden, aber beim physischen Ausfall des Loaders muss eine Ausnahme des Typs "LoaderNotAvailableException" ausgelöst werden. ObjectGrid behandelt diese beiden Ausnahmen auf unterschiedliche Weise:

- Wenn der Write-behind-Loader eine Ausnahme vom Typ "LoaderException" abfängt, geht er von einem Datenfehler aus, z. B. von einem doppelten Schlüssel. Der Write-behind-Loader löst den Aktualisierungsstapel auf und versucht, einen Datensatz nach dem anderen zu aktualisieren, um den Datenfehler zu isolieren. Wird bei dieser Aktualisierung auf Datensatzbasis erneut eine Ausnahme vom Typ "LoaderException" abgefangen, wird ein Datensatz zur fehlgeschlagenen Aktualisierung erstellt und in der Map für fehlgeschlagene Aktualisierungen protokolliert.
- Wenn der Write-behind-Loader eine Ausnahme vom Typ "LoaderNotAvailableException" abfängt, geht er von einem Ausfall aus, weil er keine Verbindung zum Datenbank-Back-End herstellen kann, z. B., weil das Datenbank-Back-End inaktiv ist, keine Datenbankverbindung verfügbar oder das Netz inaktiv ist. Der Write-behind-Loader wartet 15 Sekunden und versucht dann erneut, die Datenbankaktualisierung im Stapelbetrieb durchzuführen.

Häufig wird der Fehler gemacht, eine Ausnahme vom Typ "LoaderException" auszulösen, obwohl eigentlich eine Ausnahme vom Typ "LoaderNotAvailableException" ausgelöst werden müsste. Alle Datensätze, die in die Warteschlange für den Write-behind-Loader eingereiht sind, werden als Datensätze für eine fehlgeschlagene Aktualisierung markiert, was den eigentlich Zweck der Isolierung von Back-End-Fehlern zunichte macht.

Leistungsaspekte

Die Unterstützung des Write-behind-Cachings erhöht die Antwortzeiten, weil die Loader-Aktualisierung aus der Transaktion entfernt wird. Außerdem erhöht sich der Datenbankdurchsatz, weil Datenbankaktualisierungen kombiniert werden. Es ist wichtig, die Kosten zu kennen, die durch den Write-behind-Thread anfallen, der die Daten aus der Warteschlangen-Map extrahiert und mit Push an den Loader überträgt.

Die maximale Aktualisierungsanzahl und die maximale Aktualisierungszeit müssen den erwarteten Verwendungsmustern und der Umgebung entsprechend angepasst werden. Wenn der Wert für die maximale Aktualisierungsanzahl oder der Wert für die maximale Aktualisierungszeit zu klein gewählt wird, kann der Write-behind-Threads mehr Kosten verursachen, als er Vorteile bringt. Wenn ein sehr hoher Wert für diese beiden Parameter festgelegt wird, ist es möglich, dass die Speicherbelegung aufgrund der Einreihung der Daten zunimmt und veraltete Datensätze länger in der Datenbank verbleiben.

Um die beste Leistung zu erzielen, sollten Sie bei der Optimierung der Write-behind-Parameter die folgenden Faktoren berücksichtigen:

- Verhältnis zwischen Lese- und Schreibtransaktionen

- Aktualisierungsintervall für dieselben Datensätze
- Latenzzeit für Datenbankaktualisierung

Unterstützung des Write-behind-Cachings

Sie können Write-behind-Caching verwenden, um die Kosten für die Aktualisierung einer Back-End-Datenbank zu reduzieren. Beim Write-behind-Caching werden Aktualisierungen für das Loader-Plug-in in die Warteschlange eingereiht.

Einführung

Beim Write-behind-Caching werden Aktualisierungen für das Loader-Plug-in asynchron in die Warteschlange eingereiht. Sie können die Leistung von Aktualisierungs-, Einfüge- und Entfernungsoperationen für die Map verbessern, indem Sie die eXtreme-Scale-Transaktion von der Datenbanktransaktion entkoppeln. Die asynchrone Aktualisierung wird nach einer zeitbasierten Verzögerung (z. B. fünf Minuten) oder einer eintragsbasierten Verzögerung (z. B. 1000 Einträge) durchgeführt.

Wenn Sie die Write-behind-Einstellung in einer BackingMap konfigurieren, wird ein Write-behind-Thread erstellt und der konfigurierte Loader eingeschlossen. Wenn eine eXtreme-Scale-Transaktion einen Eintrag in einer eXtreme-Scale-Map einfügt, aktualisiert oder entfernt, wird ein LogElement-Objekt für jeden dieser Datensätze erstellt. Diese Elemente werden an den Write-behind-Loader gesendet und in eine spezielle ObjectMap, eine so genannte Warteschlangen-Map, eingereiht. Jede BackingMap mit aktivierter Write-behind-Einstellung hat ihre eigenen Warteschlangen-Maps. Ein Write-behind-Thread entfernt die in die Warteschlange eingereihten Daten aus den Warteschlangen-Maps und überträgt Sie mit Push in den echten Back-End-Loader.

Der Write-behind-Loader sendet nur LogElement-Objekte der Typen "insert" (Einfügen), "update" (Aktualisieren) und "delete" (Löschen) an den echten Loader. Alle anderen Typen von LogElement-Objekten, wie z. B. EVICT, werden ignoriert.

Die Write-behind-Unterstützung ist eine Erweiterung des Loader-Plug-ins, das Sie verwenden, um eXtreme Scale mit der Datenbank zu integrieren. Sehen Sie sich beispielsweise die Informationen zur Konfiguration eines JPA-Loaders im Abschnitt „JPA-Loader konfigurieren” auf Seite 235 an.

Vorteile

Das Aktivieren der Write-behind-Unterstützung hat die folgenden Vorteile:

- Isolation von Back-End-Fehlern: Durch das Write-behind-Caching können Back-End-Fehler isoliert werden. Wenn die Back-End-Datenbank ausfällt, werden Aktualisierungen in die Warteschlangen-Map eingereiht. Die Anwendungen können weiterhin Transaktionen an eXtreme Scale senden. Nach der Wiederherstellung des Back-Ends werden die Daten in der Warteschlangen-Map mit Push an das Back-End übertragen.
- Geringere Back-End-Last: Der Write-behind-Loader fasst die Aktualisierungen auf Schlüsselbasis so zusammen, dass nur eine einzige zusammenfasste Aktualisierung pro Schlüssel in der Warteschlangen-Map vorhanden ist. Bei dieser Zusammenfassung verringert sich die Anzahl der Aktualisierungen für das Back-End.
- Verbesserte Transaktionsleistung: Die Zeiten einzelner eXtreme-Scale-Transaktionen verringern sich, weil sie nicht auf die Synchronisation der Daten mit dem Back-End warten müssen.

ObjectGrid-XML-Deskriptordatei

Wenn Sie eXtreme Scale mit einer eXtreme-Scale-XML-Deskriptordatei konfigurieren, wird der Write-behind-Loader über das Attribut "writeBehind" im Tag "backingMap" aktiviert. Es folgt ein Beispiel:

```
<objectGrid name="library" >
     <backingMap name="book" writeBehind="T300;C900" pluginCollectionRef="bookPlugins"/>
```

Im vorherigen Beispiel wird die Write-behind-Unterstützung für die BackingMap "book" mit dem Parameter "T300;C900" aktiviert.

Das Attribut "writeBehind" gibt die maximal zulässige Aktualisierungszeit und/oder eine maximale Anzahl an Schlüsselaktualisierungen an. Der Parameter "writebehind" hat das folgende Format:

```
Attribut "writeBehind" ::= <Standardwerte> | <Aktualisierungszeit> | <Schlüsselaktualisierungsanzahl> |
<Aktualisierungszeit> ";" <Schlüsselaktualisierungsanzahl>
Aktualisierungszeit  ::= "T" <positive ganze Zahl>
Schlüsselaktualisierungsanzahl ::= "C" <positive ganze Zahl>
defaults  ::= "" {table}
```

Aktualisierungen im Loader finden statt, wenn eines der folgenden Ereignisse eintritt:

1. Die maximale Aktualisierungszeit in Sekunden seit der letzten Aktualisierung ist abgelaufen.
2. Die Anzahl aktualisierter Schlüssel in der Warteschlangen-Map hat die maximal zulässige Anzahl an Schlüsselaktualisierungen erreicht.

Diese Parameter sind lediglich Hinweise. Der echte Aktualisierungszähler und die echte Aktualisierungszeit liegen nah bei den Parametern. Es ist jedoch nicht garantiert, dass der echte Aktualisierungszähler und die echte Aktualisierungszeit den definierten Parametern entsprechen. Außerdem könnte die erste Write-behind-Aktualisierung erst nach zwei Aktualisierungszeitintervallen stattfinden. Dies ist darauf zurückzuführen, dass eXtreme Scale die Startzeit für die Aktualisierung zufällig wählt, so dass nicht alle Partitionen gleichzeitig auf die Datenbank zugreifen.

Im vorherigen Beispiel (T300;C900) schreibt der Loader die Daten 300 Sekunden nach der letzten Aktualisierung bzw. bei 900 zu aktualisierenden Schlüsseln in die Datenbank.

Die Standardaktualisierungszeit sind 300 Sekunden, und die Standardanzahl der Schlüsselaktualisierungen ist 1000.

In der folgenden Tabelle sind einige Beispiele für Write-behind-Attribute aufgelistet.

Anmerkung: Wenn Sie den Write-behind-Loader als leere Zeichenfolge konfigurieren (`writeBehind=""`), wird der Write-behind-Loader mit den Standardwerten aktualisiert. Geben Sie das Attribut "writeBehind" nicht an, wenn die Write-behind-Unterstützung nicht aktiviert werden soll.

Tabelle 8. Write-behind-Optionen

Attributwert	Zeit
T100	Die Aktualisierungszeit sind 100 Sekunden, und die Anzahl der Schlüsselaktualisierungen ist 1000 (Standardwert).
C2000	Die Aktualisierungszeit sind 300 Sekunden (Standardwert), und die Anzahl der Schlüsselaktualisierungen ist 2000.
T300;C900	Die Aktualisierungszeit sind 300 Sekunden, und die Anzahl der Schlüsselaktualisierungen ist 900.
""	Die Aktualisierungszeit sind 300 Sekunden (Standardwert), und die Anzahl der Schlüsselaktualisierungen ist 1000 (Standardwert).

Write-behind-Unterstützung programmgesteuert aktivieren

Wenn Sie eine BackingMap für eine lokale speicherinterne eXtreme-Scale-Instanz programmgesteuert erstellen, können Sie die folgende Methode in der Schnittstelle "BackingMap" verwenden, um die Write-behind-Unterstützung zu aktivieren und zu inaktivieren.

```
public void setWriteBehind(String writeBehindParam);
```

Weitere Einzelheiten hierzu finden Sie in den Informationen zur Schnittstelle BackingMap im *Programmierhandbuch*.

Hinweise zum Anwendungsdesign

Das Aktivieren der Write-behind-Unterstützung ist zwar einfach, aber eine Anwendung mit Write-behind-Unterstützung zu entwerfen, bedarf sorgfältiger Überlegungen. Ohne Write-behind-Unterstützung ist die Back-End-Transaktion in die eXtreme-Scale-Transaktion eingeschlossen. Die eXtreme-Scale-Transaktion wird vor der Back-End-Transaktion gestartet und endet erst nach Abschluss der Back-End-Transaktion.

Wenn die Write-behind-Unterstützung aktiviert ist, endet die eXtreme-Scale-Transaktion vor dem Start der Back-End-Transaktion. Die eXtreme-Scale-Transaktion und die Back-End-Transaktion sind entkoppelt.

Referenzielle Integritätsbedingungen

Jede BackingMap, die mit Write-behind-Unterstützung konfiguriert ist, hat einen eigenen Write-behind-Thread, der die Daten mit Push an das Back-End überträgt. Deshalb werden die Daten, die in einer einzigen eXtreme-Scale-Transaktion in verschiedenen Maps aktualisiert wurden, im Back-End in verschiedenen Back-End-Transaktionen aktualisiert. Beispiel: Transaktion T1 aktualisiert den Schlüssel "key1" in der Map "Map1" und den Schlüssel "key2" in der Map "Map2". Die Aktualisierungen von Schlüssel key1 in der Map Map1 und von Schlüssel "key2" in der Map "Map2" werden in einer jeweils anderen Back-End-Transaktion von einem jeweils anderen Write-behind-Thread durchgeführt. Wenn es Beziehungen zwischen den in Map1 und Map2 gespeicherten Daten, wie z. B. Integritätsbedingungen über Fremdschlüssel, im Back-End gibt, können die Aktualisierungen fehlschlagen.

Beim Design der referenziellen Integritätsbedingungen in Ihrer Back-End-Datenbank müssen Sie sicherstellen, dass solche nicht ausführbaren Aktualisierungen zugelassen werden.

Fehlgeschlagene Aktualisierungen

Da die eXtreme-Scale-Transaktion vor dem Start der Back-End-Transaktion beendet wird, ist es möglich, dass eine erfolgreiche Transaktion berichtet wird, obwohl dies in Wirklichkeit nicht der Fall ist. Wenn Sie beispielsweise versuchen, einen Eintrag in eine eXtreme-Scale-Transaktion einzufügen, die nicht in der BackingMap, aber im Back-End vorhanden ist führt dies zwar zu einem doppelten Schlüssel, aber die eXtreme-Scale-Transaktion ist trotzdem erfolgreich. Die Transaktion, in der der Write-behind-Thread das Objekt in das Back-End einfügt, scheitert jedoch mit einer Ausnahme vom Typ "Schlüssel doppelt vorhanden".

Informationen zur Behandlung solcher Fehler finden Sie im Abschnitt „Behandlung fehlgeschlagener Write-behind-Aktualisierungen” auf Seite 133.

Sperrverhalten von Warteschlangen-Maps

Ein weiterer wichtiger Unterschied im Transaktionsverhalten ist das Sperrverhalten. eXtreme Scale unterstützt drei verschiedene Sperrstrategien: PESSIMISTIC (Pessimistisch), OPTIMISITIC (Optimistisch) und NONE (Keine). Die Write-behind-Warteschlangen-Map verwendet die pessimistische Sperrstrategie, unabhängig davon, welche Sperrstrategie für die zugehörige BackingMap konfiguriert ist. Es gibt zwei verschiedene Typen von Operationen, die eine Sperre für die Warteschlangen-Map anfordern:

- Wenn eine eXtreme-Scale-Transaktion festgeschrieben wird oder eine Flush-Operation (Map-Flush oder Sitzungs-Flush) stattfindet, liest die Transaktion den Schlüssel in der Warteschlangen-Map und setzt eine S-Sperre für den Schlüssel.
- Wenn eine eXtreme-Scale-Transaktion festgeschrieben wird, versucht die Transaktion die S-Sperre für den Schlüssel in eine X-Sperre zu aktualisieren.

Anhand dieses zusätzlichen Verhaltens für die Warteschlangen-Map sind einige Unterschiede im Sperrverhalten erkennbar.

- Wenn die Benutzer-Map mit einer pessimistischen Sperrstrategie konfiguriert ist, sind die Unterschiede im Sperrverhalten nicht gravierend. Bei jedem Aufruf einer Flush- oder Festschreiboperation (Commit) wird eine S-Sperre für denselben Schlüssel in der Warteschlangen-Map gesetzt. Während der Festschreibung wird nicht nur eine X-Sperre für den Schlüssel in der Benutzer-Map, sondern auch für den Schlüssel in der Warteschlangen-Map angefordert.
- Wenn die Benutzer-Map mit einer optimistischen Sperrstrategie oder ohne Sperrstrategie konfiguriert ist, folgt die Benutzertransaktion dem Muster der pessimistischen Sperrstrategie. Bei jedem Aufruf einer Flush- oder Festschreiboperation (Commit) wird eine S-Sperre für denselben Schlüssel in der Warteschlangen-Map angefordert. Während der Festschreibung wird in derselben Transaktion eine X-Sperre für den Schlüssel in der Warteschlangen-Map angefordert.

Transaktionswiederholungen im Loader

WebSphere eXtreme Scale unterstützt keine zweiphasigen Transaktionen und keine XA-Transaktionen. Der Write-behind-Thread entfernt Datensätze aus der Warteschlangen-Map und aktualisiert die Datensätze im Back-End. Wenn der Server mitten in der Transaktion ausfällt, können einige Back-End-Aktualisierungen verloren gehen.

Der Write-behind-Loader versucht automatisch, fehlgeschlagene Transaktionen erneut zu schreiben, und sendet eine unbestätigte Protokollfolge an das Back-End, um einen Datenverlust zu verhindern. Diese Aktion erfordert, dass der Loader idempotent ist, d. h., wenn die Methode "Loader.batchUpdate(TxId, LogSequence)" zweimal mit demselben Wert aufgerufen wird, liefen diese Aufrufe dasselbe Ergebnis wie ein einmaliger Aufruf. Loader-Implementierungen müssen zum Aktivieren dieses Features die Schnittstelle "RetryableLoader" implementieren. Weitere Einzelheiten finden Sie in der API-Dokumentation.

Ausfall des Loaders

Das Loader-Plug-in kann ausfallen, wenn es nicht mit dem Datenbank-Back-End kommunizieren kann. Dies kann passieren, wenn der Datenbankserver oder die Netzverbindung inaktiv ist. Der Write-behind-Loader reiht die Aktualisierungen in

eine Warteschlange ein und versucht anschließend in regelmäßigen Abständen, die Datenänderungen mit Push an den Loader zu übertragen. Der Loader muss die Laufzeitumgebung von WebSphere eXtreme Scale darüber benachrichtigen, dass ein Problem mit der Datenbankkonnektivität vorliegt, indem er eine Ausnahme vom Typ "LoaderNotAvailableException" auslöst.

Deshalb muss die Loader-Implementierung in der Lage sein, einen Datenfehler von einem physischen Ausfall des Loaders zu unterscheiden. Bei Datenfehlern muss eine Ausnahme des Typs "LoaderException" oder "OptimisticCollisionException" ausgelöst bzw. erneut ausgelöst werden, aber beim physischen Ausfall des Loaders muss eine Ausnahme des Typs "LoaderNotAvailableException" ausgelöst werden. WebSphere eXtreme Scale behandelt diese beiden Ausnahmen auf unterschiedliche Weise:

- Wenn der Write-behind-Loader eine Ausnahme vom Typ "LoaderException" abfängt, geht er von einem Datenfehler aus, z. B. von einem doppelten Schlüssel. Der Write-behind-Loader löst den Aktualisierungsstapel auf und versucht, einen Datensatz nach dem anderen zu aktualisieren, um den Datenfehler zu isolieren. Wird bei dieser Aktualisierung auf Datensatzbasis erneut eine Ausnahme vom Typ "LoaderException" abgefangen, wird ein Datensatz zur fehlgeschlagenen Aktualisierung erstellt und in der Map für fehlgeschlagene Aktualisierungen protokolliert.
- Wenn das Write-Behind-Ladeprogramm eine Ausnahme vom Typ "LoaderNotAvailableException" abfängt, geht es von einem Ausfall aus, weil es keine Verbindung zum Datenbank-Back-End herstellen kann, z. B., weil das Datenbank-Back-End inaktiv ist, keine Datenbankverbindung verfügbar oder das Netz inaktiv ist. Der Write-behind-Loader wartet 15 Sekunden und versucht dann erneut, die Datenbankaktualisierung im Stapelbetrieb durchzuführen.

Häufig wird der Fehler gemacht, eine Ausnahme vom Typ "LoaderException" auszulösen, obwohl eigentlich eine Ausnahme vom Typ "LoaderNotAvailableException" ausgelöst werden müsste. Alle Datensätze, die in die Warteschlange für den Write-behind-Loader eingereiht sind, werden als Datensätze für eine fehlgeschlagene Aktualisierung markiert, was den eigentlich Zweck der Isolierung von Back-End-Fehlern zunichte macht. Dieser Fehler ist wahrscheinlich, wenn Sie einen generischen Loader für die Kommunikation mit Datenbanken schreiben.

Der von eXtreme Scale bereitgestellte JPALoader ist ein Beispiel. Der JPALoader verwendet die JPA-API für die Interaktion mit Datenbank-Back-Ends. Wenn das Netz ausfällt, empfängt der JPALoader eine Ausnahme des Typs "javax.persistence.PersitenceException", aber er weiß nichts über den Schweregrad des Fehlers, sofern der SQL-Status- und -Fehlercode der verketteten SQLException nicht geprüft werden. Die Tatsache, dass der JPALoader für alle Typen von Datenbanken verwendet werden kann, macht das Problem noch komplexer, da die SQL-Status und -Fehlercodes für Netzausfallprobleme anders sind. Für die Lösung dieses Problems stellt WebSphere eXtreme Scale eine API "ExceptionMapper" bereit, damit Benutzer eine Implementierung integrieren können, um eine Ausnahme einer aussagefähigeren Ausnahme zuordnen zu können. Benutzer können beispielsweise eine generische Ausnahme "javax.persistence.PersitenceException" einer Ausnahme "LoaderNotAvailableException" zuordnen, wenn der SQL-Status -bzw. -Fehlercode darauf hinweist, dass das Netz ausgefallen ist.

Leistungsaspekte

Die Unterstützung des Write-behind-Cachings erhöht die Antwortzeiten, weil die Loader-Aktualisierung aus der Transaktion entfernt wird. Außerdem erhöht sich

der Datenbankdurchsatz, weil Datenbankaktualisierungen kombiniert werden. Es ist wichtig, die Kosten zu kennen, die durch den Write-behind-Thread anfallen, der die Daten aus der Warteschlangen-Map extrahiert und mit Push an den Loader überträgt.

Die maximale Aktualisierungsanzahl und die maximale Aktualisierungszeit müssen den erwarteten Verwendungsmustern und der Umgebung entsprechend angepasst werden. Wenn der Wert für die maximale Aktualisierungsanzahl oder der Wert für die maximale Aktualisierungszeit zu klein gewählt wird, kann der Write-behind-Threads mehr Kosten verursachen, als er Vorteile bringt. Wenn ein sehr hoher Wert für diese beiden Parameter festgelegt wird, ist es möglich, dass die Speicherbelegung aufgrund der Einreihung der Daten zunimmt und veraltete Datensätze länger in der Datenbank verbleiben.

Um die beste Leistung zu erzielen, sollten Sie bei der Optimierung der Write-behind-Parameter die folgenden Faktoren berücksichtigen:

- Verhältnis zwischen Lese- und Schreibtransaktionen
- Aktualisierungsintervall für dieselben Datensätze
- Latenzzeit für Datenbankaktualisierung

Behandlung fehlgeschlagener Write-behind-Aktualisierungen

Da die Transaktion von WebSphere eXtreme Scale vor dem Start der Back-End-Transaktion beendet wird, ist es möglich, dass eine erfolgreiche Transaktion berichtet wird, obwohl dies in Wirklichkeit nicht der Fall ist.

Wenn Sie beispielsweise versuchen, einen Eintrag in eine eXtreme-Scale-Transaktion einzufügen, die nicht in der BackingMap, aber im Back-End vorhanden ist führt dies zwar zu einem doppelten Schlüssel, aber die eXtreme-Scale-Transaktion ist trotzdem erfolgreich. Die Transaktion, in der der Write-behind-Thread das Objekt in das Back-End einfügt, scheitert jedoch mit einer Ausnahme vom Typ "Schlüssel doppelt vorhanden".

Behandlung fehlgeschlagener Write-behind-Aktualisierungen: Clientseite

Eine solche Aktualisierung und jede andere fehlgeschlagene Back-End-Aktualisierung ist eine fehlgeschlagene Write-behind-Aktualisierung. Fehlgeschlagene Write-behind-Aktualisierungen werden in einer Map für fehlgeschlagene Write-behind-Aktualisierungen gespeichert. Diese Map dient als Ereigniswarteschlange für fehlgeschlagene Aktualisierungen. Der Schlüssel der Aktualisierung ist ein eindeutiges Integer-Objekt, und der Wert ist eine Instanz von FailedUpdateElement. Die fehlgeschlagene Write-behind-Aktualisierungs-Map ist mit einem Evictor (Bereinigungsprogramm) konfiguriert, der die Datensätze eine Stunde nach Einfügen entfernt. Deshalb gehen Datensätze der fehlgeschlagenen Aktualisierung verloren, wenn sie nicht innerhalb von einer Stunde abgerufen werden.

Mit der API "ObjectMap" können die Map-Einträge für fehlgeschlagene Write-behind-Aktualisierung abgerufen werden. Die Map für fehlgeschlagene Write-behind-Aktualisierungen hat den Namen "IBM_WB_FAILED_UPDATES_<Map-Name>". Die Präfixnamen für die einzelnen Write-behind-System-Maps finden Sie in der Dokumentation zur API "WriteBehindLoaderConstants". Es folgt ein Beispiel.

```
Prozess fehlgeschlagen - Mustercode
ObjectMap failedMap = session.getMap(
    WriteBehindLoaderConstants.WRITE_BEHIND_FAILED_UPDATES_MAP_PREFIX + "Employee");
Object key = null;

session.begin();
```

```
while(key = failedMap.getNextKey(ObjectMap.QUEUE_TIMEOUT_NONE)) {
    FailedUpdateElement element = (FailedUpdateElement) failedMap.get(key);
    Throwable throwable = element.getThrowable();
    Object failedKey = element.getKey();
    Object failedValue = element.getAfterImage();
    failedMap.remove(key);
    // Schlüssel, Wert oder Ausnahme verarbeiten

}
session.commit();
```

Ein getNextKey-Aufruf arbeitet für jede eXtreme-Scale-Transaktion mit einer bestimmten Partition. In einer verteilten Umgebung müssen Sie zum Abrufen der Schlüssel aus allen Partitionen mehrere Transaktionen starten, wie im folgenden Beispiel gezeigt wird:

Schlüssel von allen Partitionen abrufen - Mustercode

```
ObjectMap failedMap = session.getMap(
    WriteBehindLoaderConstants.WRITE_BEHIND_FAILED_UPDATES_MAP_PREFIX + "Employee");
while (true) {
    session.begin();
    Object key = null;
    while(( key = failedMap.getNextKey(5000) )!= null ) {
        FailedUpdateElement element = (FailedUpdateElement) failedMap.get(key);
    Throwable throwable = element.getThrowable();
        Object failedKey = element.getKey();
        Object failedValue = element.getAfterImage();
        failedMap.remove(key);
        // Schlüssel, Wert oder Ausnahme verarbeiten

    }
    Session.commit();
}
```

Anmerkung: Die Map für fehlgeschlagene Aktualisierungen ist eine Möglichkeit, die Vitalität (den ordnungsgemäßen Betrieb) der Anwendung zu überwachen. Wenn ein System sehr viele Datensätze in der Map für fehlgeschlagene Aktualisierungen erzeugt, ist dies ein Hinweis darauf, dass die Anwendung bzw. Architektur überprüft bzw. so überarbeitet werden sollte, dass die Write-behind-Unterstützung genutzt wird. Ab Version 6.1.0.5 können Sie das xsadmin-Script verwenden, um die Größe von Einträgen in der Map für fehlgeschlagene Aktualisierungen zu ermitteln.

Behandlung fehlgeschlagener Write-behind-Aktualisierungen: Shard-Listener

Eine fehlgeschlagene Write-behind-Transaktion sollte unbedingt erkannt und protokolliert werden. Jede Anwendung, die die Write-behind-Technik verwendet, muss einen Watcher (Überwachungsprogramm) für die Behandlung fehlgeschlagener Write-behind-Aktualisierungen implementieren. Auf diese Weise kann ein Speicherengpass verhindert werden, wenn Datensätze in der Map für fehlgeschlagene Aktualisierungen nicht entfernt werden, weil die Bereinigung durch die Anwendung erwartet wird.

Der folgende Code veranschaulicht, wie ein solcher Watcher oder Dumper integriert wird, der wie im folgenden Snippet der ObjectGrid-Deskriptor-XML hinzugefügt werden muss:

```
<objectGrid name="Grid">
         <bean id="ObjectGridEventListener" className="utils.WriteBehindDumper"/>
```

Wie Sie sehen, wurde die die Bean "ObjectGridEventListener" hinzugefügt. Diese Bean ist der zuvor erwähnte Write-behind-Watcher. Der Watcher interagiert mit

den Maps für alle primären Shards in einer JVM und sucht nach denen, in denen die Write-behind-Technik aktiviert ist. Wenn er ein solches Shard findet, versucht er, bis zu 100 fehlgeschlagene Aktualisierungen zu protokollieren. Er überwacht ein primäres Shard so lange, bis das Shard in eine andere JVM verschoben wird. Alle Anwendungen, die die Write-behind-Technik verwenden, *müssen* einen Watcher verwenden, der diesem gleicht. Andernfalls kann in den Java Virtual Machines ein Speicherengpass auftreten, weil die Fehler-Map nie bereinigt wird.

Weitere Informationen finden Sie im Abschnitt "Mustercodes für eine Write-behind-Dumper-Klasse".

Mustercode für eine Write-behind-Dumper-Klasse

Dieser Musterquellcode veranschaulicht, wie Sie einen Watcher (Dumper) für die Behandlung fehlgeschlagener Write-behind-Aktualisierungen schreiben.

```
//
// Dieses Musterprogramm wird ohne Wartung (auf "as-is"-Basis)
// bereitgestellt und kann vom Kunden (a) zu Schulungs- und Studienzwecken,
// (b) zum Entwickeln von Anwendungen für ein IBM WebSphere-Produkt zur
// internen Nutzung beim Kunden oder Weitergabe im Rahmen einer solchen
// Anwendung in kundeneigenen Produkten gebührenfrei genutzt, ausgeführt,
// kopiert und geändert werden."
//
//5724-J34 (C) COPYRIGHT International Business Machines Corp. 2009
//Alle Rechte vorbehalten * Lizenziertes Material - Eigentum von IBM
//
package utils;

import java.util.Collection;
import java.util.Iterator;
import java.util.concurrent.Callable;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.ScheduledFuture;
import java.util.concurrent.ScheduledThreadPoolExecutor;
import java.util.concurrent.TimeUnit;
import java.util.logging.Logger;

import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridException;
import com.ibm.websphere.objectgrid.ObjectGridRuntimeException;
import com.ibm.websphere.objectgrid.ObjectMap;
import com.ibm.websphere.objectgrid.Session;
import com.ibm.websphere.objectgrid.UndefinedMapException;
import com.ibm.websphere.objectgrid.plugins.ObjectGridEventGroup;
import com.ibm.websphere.objectgrid.plugins.ObjectGridEventListener;
import com.ibm.websphere.objectgrid.writebehind.FailedUpdateElement;
import com.ibm.websphere.objectgrid.writebehind.WriteBehindLoaderConstants;

/**
 * Write behind expects transactions to the Loader to succeed. If a transaction for a key fails then
 * it inserts an entry in a Map called PREFIX + mapName. The application should be checking this
 * map for entries to dump out write behind transaction failures. The application is responsible for
 * analyzing and then removing these entries. These entries can be large as they include the key, before
 * and after images of the value and the exception itself. Exceptions can easily be 20k on their own.
 *
 * The class is registered with the grid and an instance is created per primary shard in a JVM. It creates
 * a single thread
 * and that thread then checks each write behind error map for the shard, prints out the problem and
 * then removes the entry.
 *
 * This means there will be one thread per shard. If the shard is moved to another JVM then the deactivate
 * method stops the thread.
 * @author bnewport
 *
 */
public class WriteBehindDumper implements ObjectGridEventListener, ObjectGridEventGroup.ShardEvents, Callable<Boolean>
{
 static Logger logger = Logger.getLogger(WriteBehindDumper.class.getName());

 ObjectGrid grid;

 /**
  * Thread pool to handle table checkers. If the application has it's own pool
  * then change this to reuse the existing pool
  */
 static ScheduledExecutorService pool = new ScheduledThreadPoolExecutor(2); // two threads to dump records

 // the future for this shard
 ScheduledFuture<Boolean> future;

 // true if this shard is active
 volatile boolean isShardActive;
```

```
/**
 * Normal time between checking Maps for write behind errors
 */
final long BLOCKTIME_SECS = 20L;

/**
 * An allocated session for this shard. No point in allocating them again and again
 */
Session session;
/**
 * When a primary shard is activated then schedule the checks to periodically check
 * the write behind error maps and print out any problems
 */
public void shardActivated(ObjectGrid grid) {
 try
 {
  this.grid = grid;
  session = grid.getSession();

  isShardActive = true;
  future = pool.schedule(this, BLOCKTIME_SECS, TimeUnit.SECONDS); // check every BLOCKTIME_SECS seconds initially
 }
 catch(ObjectGridException e)
 {
  throw new ObjectGridRuntimeException("Exception activating write dumper", e);
 }
}

/**
 * Mark shard as inactive and then cancel the checker
 */
public void shardDeactivate(ObjectGrid arg0)
{
 isShardActive = false;
 // if it's cancelled then cancel returns true
 if(future.cancel(false) == false)
 {
  // otherwise just block until the checker completes
  while(future.isDone() == false) // wait for the task to finish one way or the other
  {
   try
   {
    Thread.sleep(1000L); // check every second
   }
   catch(InterruptedException e)
   {
   }
  }
 }
}

/**
 * Simple test to see if the map has write behind enabled and if so then return
 * the name of the error map for it.
 * @param mapName The map to test
 * @return The name of the write behind error map if it exists otherwise null
 */
static public String getWriteBehindNameIfPossible(ObjectGrid grid, String mapName)
{
 BackingMap map = grid.getMap(mapName);
 if(map != null && map.getWriteBehind() != null)
 {
  return WriteBehindLoaderConstants.WRITE_BEHIND_FAILED_UPDATES_MAP_PREFIX + mapName;
 }
 else
        return null;
}

/**
 * This runs for each shard. It checks if each map has write behind enabled and if it does
  * then it prints out any write behind
 * transaction errors and then removes the record.
 */
public Boolean call()
{
 logger.fine("Called for " + grid.toString());
 try
 {
  // while the primary shard is present in this JVM
  // only user defined maps are returned here, no system maps like write behind maps are in
  // this list.
  Iterator<String> iter = grid.getListOfMapNames().iterator();
  boolean foundErrors = false;
  // iterate over all the current Maps
  while(iter.hasNext() && isShardActive)
  {
   String origName = iter.next();

   // if it's a write behind error map
   String name = getWriteBehindNameIfPossible(grid, origName);
```

```
   if(name != null)
   {
    // try to remove blocks of N errors at a time
    ObjectMap errorMap = null;
    try
    {
     errorMap = session.getMap(name);
    }
    catch(UndefinedMapException e)
    {
     // at startup, the error maps may not exist yet, patience...
     continue;
    }
    // try to dump out up to N records at once
    session.begin();
    for(int counter = 0; counter < 100; ++counter)
    {
     Integer seqKey = (Integer)errorMap.getNextKey(1L);
     if(seqKey != null)
     {
      foundErrors = true;
      FailedUpdateElement elem = (FailedUpdateElement)errorMap.get(seqKey);
      //
      // Your application should log the problem here
      logger.info("WriteBehindDumper ( " + origName + ") for key (" + elem.getKey() + ") Exception: " +
       elem.getThrowable().toString());
      //
      //
      errorMap.remove(seqKey);
     }
     else
      break;
    }
    session.commit();
   }
  } // do next map
  // loop faster if there are errors
  if(isShardActive)
  {
   // reschedule after one second if there were bad records
   // otherwise, wait 20 seconds.
   if(foundErrors)
    future = pool.schedule(this, 1L, TimeUnit.SECONDS);
   else
    future = pool.schedule(this, BLOCKTIME_SECS, TimeUnit.SECONDS);
  }
 }
 catch(ObjectGridException e)
 {
  logger.fine("Exception in WriteBehindDumper" + e.toString());
  e.printStackTrace();

  //don't leave a transaction on the session.
  if(session.isTransactionActive())
  {
   try { session.rollback(); } catch(Exception e2) {}
  }
 }
 return true;
}

public void destroy() {
 // TODO Auto-generated method stub

}

public void initialize(Session arg0) {
 // TODO Auto-generated method stub

}

public void transactionBegin(String arg0, boolean arg1) {
 // TODO Auto-generated method stub

}

public void transactionEnd(String arg0, boolean arg1, boolean arg2,
  Collection arg3) {
 // TODO Auto-generated method stub

}
}
```

Peer-to-Peer-Replikation mit JMS konfigurieren

Der JMS-basierte (Java Message Service) Peer-to-Peer-Replikationsmechanismus wird in der verteilten und in der lokalen Umgebung von WebSphere eXtreme Scale verwendet. JMS ist ein Kern-zu-Kern-Replikationsprozess und lässt die Übertra-

gung von Datenaktualisierungen zwischen lokalen ObjectGrids und verteilten ObjectGrids zu. Mit diesem Mechanismus können Sie beispielsweise Datenaktualisierungen aus einem verteilten eXtreme-Scale-Grid in ein lokales eXtreme-Scale-Grid oder aus einem anderen Grid in einer anderen Systemdomäne verschieben.

Vorbereitende Schritte

Der JMS-basierte Peer-to-Peer-Replikationsmechanismus basiert auf den integrierten JMS-basierten Schnittstellen "ObjectGridEventListener" und "com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener". Ausführliche Informationen zum Aktivieren des Peer-to-Peer-Replikationsmechanismus finden Sie im Abschnitt „JMS-Ereignis-Listener” auf Seite 142.

Weitere Informationen finden Sie im Abschnitt „Mechanismus für Clientinaktivierung aktivieren” auf Seite 215.

Im Folgenden sehen Sie ein XML-Konfigurationsbeispiel für die Aktivierung eines Peer-to-Peer-Replikationsmechanismus in einer eXtreme-Scale-Konfiguration:

Konfiguration der Peer-to-Peer-Replikation - XML-Beispiel

```
<bean id="ObjectGridEventListener"
className="com.ibm.websphere.objectgrid.plugins.JMSObjectGridEventListener">
 <property name="replicationRole" type="java.lang.String" value="DUAL_ROLES" description="" />
  <property name="replicationStrategy" type="java.lang.String" value="PUSH" description="" />
      <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String"
  value="defaultTCF" description="" />
  <property name="jms_topicJndiName" type="java.lang.String" value="defaultTopic" description="" />
  <property name="jms_topicName" type="java.lang.String" value="defaultTopic" description="" />
  <property name="jms_userid" type="java.lang.String" value="" description="" />
  <property name="jms_password" type="java.lang.String" value="" description="" />
  <property name="jndi_properties" type="java.lang.String"
 value="java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory;
 java.naming.provider.url=tcp://localhost:61616;connectionFactoryNames=defaultTCF;
 topic.defaultTopic=defaultTopic"
 description="jndi properties" />
     </bean>
```

Änderungen an Peer-JVMs verteilen

Die Objekte "LogSequence" und "LogElement" verteilen Änderungen zwischen Peer-JVMs und kommunizieren Änderungen, die in einer eXtreme-Scale-Transaktion stattfinden über ein ObjectGridEventListener-Plug-in.

Weitere Informationen zur Verwendung von Java Message Service (JMS) für die Verteilung von Transaktionsänderungen finden Sie in den Informationen zur Verwendung von JMS für die Verteilung von Transaktionsänderungen in der *Produktübersicht*.

Eine Voraussetzung ist, dass die ObjectGrid-Instanz vom ObjectGridManager zwischengespeichert wird. Weitere Einzelheiten finden Sie in den Informationen zu den createObjectGrid-Methoden. Die boolesche Eigenschaft "cacheInstance" muss auf "true" gesetzt werden.

Dieser Mechanismus muss nicht implementiert werden. Sie können einen integrierten Mechanismus für die Peer-to-Peer-Replikation verwenden, um diese Funktion zu nutzen. Weitere Informationen finden Sie in der Dokumentation zur Peer-to-Peer-Replikation mit JMS im *Administratorhandbuch*.

Eine Anwendung kann diese Objekte verwenden, um Änderungen, die in einem ObjectGrid vorgenommen werden, problemlos über einen Nachrichtentransport an die Peer-ObjectGrids in fernen Java Virtual Machines zu veröffentlichen und die Änderungen anschließend in dieser JVM anzuwenden. Die Klasse "LogSequence-Transformer" ist für die Aktivierung dieser Unterstützung kritisch. In diesem Ab-

schnitt wird beschrieben, wie ein Listener mit einem JMS-Messaging-System für die Weitergabe der Nachrichten geschrieben wird. Zu diesem Zweck unterstützt eXtreme Scale die Übertragung von LogSequence-Objekten, die sich aus der Festschreibung einer eXtreme-Scale-Transaktion ergeben, über ein von IBM bereitgestelltes Plug-in an die Cluster-Member von WebSphere Application Server. Diese Funktion ist standardmäßig nicht aktiviert, kann aber konfiguriert werden. Wenn der Konsument oder Erzeuger jedoch kein WebSphere Application Server ist, kann die Verwendung eines externen JMS-Messaging-Systems erforderlich sein.

Mechanismus implementieren

Die Klasse "LogSequenceTransformer" und die APIs "ObjectGridEventListener", "LogSequence" und "LogElement" lassen die Verwendung jedes zuverlässigen Publish/Subscribe-Mechanismus für die Verteilung der Änderungen und die Filterung der zu verteilenden Maps zu. Die Snippets in diesem Abschnitt veranschaulichen, wie diese APIs mit JMS verwendet werden können, um ein Peer-to-Peer-ObjectGrid zu erstellen, das von Anwendungen gemeinsam genutzt wird, die sich auf verschiedenen Plattformen befinden, die einen gemeinsamen Nachrichtentransport verwenden.

Plug-in initialisieren

Das ObjectGrid ruft im Rahmen des ObjectGridEventListener-Schnittstellenvertrags die Methode "initialize" des Plug-ins auf, wenn das ObjectGrid gestartet wird. Die Methode "initialize" muss seine JMS-Ressourcen, einschließlich Verbindungen, Sitzungen und Veröffentlichungskomponenten (so genannten Publishern), anfordern und den Thread für den JMS-Listener starten.

Die folgenden Beispiele zeigen die Methode "initialize":

Beispiel für die Methode initialize

```
public void initialize(Session session) {
        mySession = session;
        myGrid = session.getObjectGrid();
        try {
            if (mode == null) {
                throw new ObjectGridRuntimeException("No mode specified");
            }
            if (userid != null) {
                connection = topicConnectionFactory.createTopicConnection(userid,
        password);
            } else
                connection = topicConnectionFactory.createTopicConnection();

            // Die Verbindung muss gestartet werden, um Nachrichten zu empfangen.
            connection.start();

            // Die JMS_Sitzung ist nicht transaktionsorientiert (false).
            jmsSession = connection.createTopicSession(false,
       javax.jms.Session.AUTO_ACKNOWLEDGE);
            if (topic == null)
                if (topicName == null) {
                    throw new ObjectGridRuntimeException("Topic not specified");
                } else {
                    topic = jmsSession.createTopic(topicName);
                }
            publisher = jmsSession.createPublisher(topic);
            // Listener-Thread starten.
            listenerRunning = true;
            listenerThread = new Thread(this);
            listenerThread.start();
```

```
        } catch (Throwable e) {
            throw new ObjectGridRuntimeException("Cannot initialize", e);
        }
    }
```

Der Code zum Starten des Threads verwendet einen Java-SE-Thread (Java 2 Platform, Standard Edition). Wenn Sie einen Server von WebSphere Application Server Version 6.x oder WebSphere Application Server Version 5.x ausführen, verwenden Sie die API für asynchrone Beans, um diesen Dämon-Thread zu starten. Sie können auch die allgemeinen APIs verwenden. Im Folgenden sehen Sie ein Beispiel für ein Ersatz-Snippet, dass diese Aktion mit einem Arbeitsmanager veranschaulicht:

```
// Listener-Thread starten.
listenerRunning = true;
workManager.startWork(this, true);
```

Das Plug-in muss die Schnittstelle "Work" an Stelle der Schnittstelle "Runnable" implementieren. Außerdem müssen Sie eine Methode "release" hinzufügen, um die Variable "listenerRunning" auf "false" zu setzen. Das Plug-in muss mit einer WorkManager-Instanz in seinem Konstruktor bzw. bei Verwendung eines IoC-Containers (Inversion of Control) durch Injektion bereitgestellt werden.

Änderungen übertragen

Im Folgenden sehen Sie eine Beispielmethode "transactionEnd" für die Veröffentlichung der lokalen Änderungen, die in einem ObjectGrid vorgenommen werden. In diesem Beispiel wird JMS verwendet, aber Sie können jeden Nachrichtentransport verwenden, der zuverlässiges Publish/Subscribe-Messaging unterstützt.

Beispiel für die Methode transactionEnd

```
     // Diese Methode wird synchronisiert, um sicherzustellen,
    // dass die Nachrichten in der Reihenfolge veröffentlicht werden, in die
    // Transaktionen festgeschrieben werden. Falls die Veröffentlichung der Nachrichten
    // parallel gestartet wird, könnten die Empfänger die Map beschädigen,
    // da Löschanforderungen vor Einfügeanforderungen usw. ankommen könnten.
public synchronized void transactionEnd(String txid, boolean isWriteThroughEnabled, boolean committed,
Collection changes) {
        try {
            // Muss Write-through und festgeschrieben sein.
            if (isWriteThroughEnabled && committed) {
                // Folgen in eine Bytefeldgruppe (byte []) schreiben.
                ByteArrayOutputStream bos = new ByteArrayOutputStream();
                ObjectOutputStream oos = new ObjectOutputStream(bos);
                if (publishMaps.isEmpty()) {
                    // Gesamte Sammlung serialisieren
                    LogSequenceTransformer.serialize(changes, oos, this, mode);
                } else {
                    // LogSequence-Objekte auf der Basis des publishMaps-Inhalts filtern.
                    Collection publishChanges = new ArrayList();
                    Iterator iter = changes.iterator();
                      while (iter.hasNext()) {
                        LogSequence ls = (LogSequence) iter.next();
                        if (publishMaps.contains(ls.getMapName())) {
                            publishChanges.add(ls);
                        }
                    }
                    LogSequenceTransformer.serialize(publishChanges, oos, this, mode);
                }
                // Objektnachricht für die Änderungen erstellen.
                oos.flush();
                ObjectMessage om = jmsSession.createObjectMessage(bos.toByteArray());
                // Eigenschaften festlegen.
                om.setStringProperty(PROP_TX, txid);
                om.setStringProperty(PROP_GRIDNAME, myGrid.getName());
                // Übertragen.
                publisher.publish(om);
            }
        } catch (Throwable e) {
            throw new ObjectGridRuntimeException("Cannot push changes", e);
        }
    }
```

In dieser Methode werden verschiedene Instanzvariablen verwendet:

- Variable `jmsSession`: Eine JMS-Sitzung, die zum Veröffentlichen von Nachrichten verwendet wird. Sie wird bei der Initialisierung des Plug-ins erstellt.
- Variable `mode`: Der Verteilungsmodus.
- Variable `publishMaps`: Eine Gruppe, die die Namen der einzelnen Maps mit zu veröffentlichenden Änderungen enthält. Wenn die Variable leer ist, werden alle Maps veröffentlicht.
- Variable `publisher`: Ein TopicPublisher-Objekt, das während der Ausführung der Methode "initialize" des Plug-ins ausgeführt wird.

Aktualisierungsnachricht empfangen und anwenden

Im Folgenden sehen Sie eine Beispielmethode "run". Diese Methode wird in einer Schleife ausgeführt, bis die Anwendung die Schleife stoppt. In jeder Schleifeniteration wird versucht, eine JMS-Nachricht zu empfangen und auf das ObjectGrid anzuwenden.

Beispiel für die Methode run für JMS-Nachrichten

```
private synchronized boolean isListenerRunning() {
        return listenerRunning;
    }

    public void run() {
        try {
            System.out.println("Listener starting");
            // JMS-Sitzung für den Empfang der Nachrichten abrufen.
            // Nicht transaktionsorientiert.
            TopicSession myTopicSession;
            myTopicSession = connection.createTopicSession(false, javax.jms.
       Session.AUTO_ACKNOWLEDGE);

            // Subskribenten für das Topic abrufen. True gibt an, dass keine Nachrichten
            // empfangen werden, die über Publisher in dieser Verbindung übertragen
            // wurden. Sonst werden eigene Aktualisierungen empfangen.
            TopicSubscriber subscriber = myTopicSession.createSubscriber(topic,
       null, true);
            System.out.println("Listener started");
            while (isListenerRunning()) {
                ObjectMessage om = (ObjectMessage) subscriber.receive(2000);
                if (om != null) {
                    // Sitzung verwenden, die bei der Initialisierung übergeben wurde.
                    // Sehr wichtig, dass Write-through (Durchschreiben) hier nicht
                    // verwendet wird.
                    mySession.beginNoWriteThrough();
                    byte[] raw = (byte[]) om.getObject();
                    ByteArrayInputStream bis = new ByteArrayInputStream(raw);
                    ObjectInputStream ois = new ObjectInputStream(bis);
                    // LogSequence-Objekte dekomprimieren.
                    Collection collection = LogSequenceTransformer.inflate(ois,
            myGrid);
                    Iterator iter = collection.iterator();
                      while (iter.hasNext()) {
                        // Änderungen jeder Map entsprechend dem bei der
                        // Serialisierung des LogSequence-Objekts verwendeten
                        // Modus verarbeiten.
                        LogSequence seq = (LogSequence) iter.next();
                        mySession.processLogSequence(seq);
                    }
                    mySession.commit();
                } // Wenn eine Nachricht vorhanden ist
            } // while loop
            // Verbindung stoppen.
            connection.close();
```

```
        } catch (IOException e) {
            System.out.println("IO Exception: " + e);
        } catch (JMSException e) {
            System.out.println("JMS Exception: " + e);
        } catch (ObjectGridException e) {
            System.out.println("ObjectGrid exception: " + e);
            System.out.println("Caused by: " + e.getCause());
        } catch (Throwable e) {
            System.out.println("Exception : " + e);
        }
        System.out.println("Listener stopped");
    }
```

JMS-Ereignis-Listener

Der JMSObjectGridEventListener unterstützt einen Mechanismus für die Inaktivierung des clientseitigen nahen Caches und einen Mechanismus für die Peer-to-Peer-Replikation. Er ist eine JMS-Implementierung (Java Message Service) der Schnittstelle "ObjectGridEventListener".

Der Mechanismus für die Clientinaktivierung kann in einer verteilten eXtreme-Scale-Umgebung verwendet werden, um sicherzustellen, dass die Daten im clientnahen Cache mit Servern oder anderen Clients synchronisiert werden. Ohne diese Funktion könnte der clientnahe Cache veraltete Daten enthalten. Aber selbst mit diesem JMS-basierten Mechanismus für Clientinaktivierung müssen Sie wegen der Verzögerung für die Laufzeitumgebung beim Veröffentlichen von Aktualisierungen das Zeitfenster für die Aktualisierung eines clientnahen Caches berücksichtigen.

Der Mechanismus für die Peer-to-Peer-Replikation kann in verteilten und lokalen eXtreme-Scale-Umgebungen verwendet werden. Er ist ein Kern-zu-Kern-Replikationsprozess und lässt die Übertragung von Datenaktualisierungen zwischen lokalen ObjectGrids und verteilten ObjectGrids zu. Mit diesem Mechanismus können Sie beispielsweise Datenaktualisierungen aus einem verteilten Grid in ein lokales ObjectGrid oder aus einem anderen Grid in einer anderen Systemdomäne verschieben.

Der JMSObjectGridEventListener erfordert, dass der Benutzer JMS- und JNDI-Informationen (Java Naming and Directory Interface) konfiguriert, damit die erforderlichen JMS-Ressourcen abgerufen werden. Außerdem müssen replikationsbezogene Eigenschaften ordnungsgemäß gesetzt werden. In einer JEE-Umgebung muss JNDI in Web- und EJB-Containern (Enterprise JavaBean) verfügbar sein. In diesem Fall ist die JNDI-Eigenschaft optional, sofern Sie keine externen JMS-Ressourcen abrufen möchten.

Dieser Ereignis-Listener hat Eigenschaften, die Sie über XML- oder programmgesteuerte Ansätze konfigurieren und nur für die Clientinaktivierung und/oder nur für die Peer-to-Peer-Replikation verwenden können. Die meisten Eigenschaften sind für die Anpassung des Verhaltens zum Erzielen der erforderlichen Funktionalität optional.

.

Weitere Informationen finden Sie in den Informationen zur API "JMSObjectGridEventListener".

JMSObjectGridEventListener-Plug-in erweitern

Das JMSObjectGridEventListener-Plug-in ermöglicht Peer-ObjectGrid-Instanzen, Aktualisierungen zu empfangen, wenn die Daten im Grid geändert oder entfernt

wurden. Außerdem ermöglicht es die Benachrichtigung von Clients, wenn Einträge aktualisiert oder einem eXtreme-Scale-Grid entfernt werden. In diesem Abschnitt wird beschrieben, wie das JMSObjectGridEventListener-Plug-in erweitert werden kann, damit Anwendungen beim Empfang einer JMS-Nachricht benachrichtigt werden können. Dies ist äußerst hilfreich, wenn die Einstellung CLIENT_SERVER-_MODEL für die Clientinaktivierung verwendet wird.

Bei der Ausführung in der Empfängerrolle wird die überschriebene Methode "JMS-ObjectGridEventListener.onMessage" automatisch von der eXtreme-Scale-Laufzeitumgebung aufgerufen, wenn die JMSObjectGridEventListener-Instanz JMS-Nachrichtenaktualisierungen vom Grid empfängt. Diese Nachrichten schließen eine Sammlung von LogSequence-Objekten ein. Die LogSequence-Objekte werden an die Methode "onMessage" übergeben, und die Anwendung verwendet das LogSequence-Objekt, um die Cacheeinträge zu identifizieren, die eingefügt, gelöscht, aktualisiert oder ungültig gemacht wurden.

Zur Verwendung des onMessage-Erweiterungspunkts führen Anwendungen die folgenden Schritte aus:

1. Erstellen Sie eine neue Klasse, die die Klasse "JMSObjectGridEventListener" erweitert und die Methode "onMessage" überschreibt.
2. Konfigurieren Sie den erweiterten JMSObjectGridEventListener auf dieselbe Weise wie den ObjectGridEventListener für ObjectGrid.

Die erweiterte Klasse "JMSObjectGridEventListener" ist eine Unterklasse der Klasse "JMSObjectGridEventListener" und kann nur zwei Methoden überschreiben: initialize (optional) und onMessage. Wenn eine Unterklasse der Klasse "JMSObjectGridEventListener" ObjectGrid-Artefakte wie ObjectGrid oder Session in der Methode "onMessage" verwenden muss, kann sie diese Artefakte in der Methode "initialize" abrufen und als Instanzvariablen zwischenspeichern. In der Methode "onMessage" können zwischengespeicherte ObjectGrid-Artefakte verwendet werden, um eine übergebene LogSequence-Sammlun zu verarbeiten.

Anmerkung: Die überschriebene Methode "initialize" muss die Methode "super.initialize" aufrufen, um den übergeordneten JMSObjectGridEventListener ordnungsgemäß zu initialisieren.

Im Folgenden sehen Sie ein Beispiel für eine erweiterte JMSObjectGridEventListener-Klasse:

```
package com.ibm.websphere.samples.objectgrid.jms.price;

import java.util.*;
import com.ibm.websphere.objectgrid.*;
import com.ibm.websphere.objectgrid.plugins.LogElement;
import com.ibm.websphere.objectgrid.plugins.LogSequence;
import com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener;

public class ExtendedJMSObjectGridEventListener extends JMSObjectGridEventListener{
 protected static boolean debug = true;

    /**
     * Dieses Grid ist dem Listener zugeordnet.
     */
     ObjectGrid grid;

    /**
     * Die Sitzung, die dem Listener zugeordnet ist
     */
     Session session;

    String objectGridType;

    public List receivedLogSequenceList = new ArrayList();
```

```
 /* (non-Javadoc)
  * @see com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener
    #initialize(com.ibm.websphere.objectgrid.Session)
  */
public void initialize(Session session) {
  // Anmerkung: Wenn Sie ein ObjectGrid-Artefakt verwenden müssen, muss diese Klasse
  // ObjectGrid von der übergebenen Session-Instanz und die ObjectMap von der
  // Session-instanz für jede transaktionsorientierte ObjectGrid-Map-Operation abrufen.
  super.initialize(session);  // Die Methode initialize der Superklasse muss aufgerufen werden
  this.session = session; // Session-Instanz zwischenspeichern, falls sie
  //   zum Durchführen der Map-Operation benötigt wird
  this.grid = session.getObjectGrid(); // ObjectGrid abrufen, falls
  //   ObjectGrid-Informationen abgerufen werden müssen
  if (grid.getObjectGridType() == ObjectGrid.CLIENT)
   objectGridType = "CLIENT";
  else if (grid.getObjectGridType() == ObjectGrid.SERVER)
   objectGridType = "Server";

  if (debug)
   System.out.println("ExtendedJMSObjectGridEventListener[" +
     objectGridType + "].initialize() : grid = " + this.grid);
 }

 /* (non-Javadoc)
  * @see com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener
    #onMessage(java.util.Collection)
  */
 protected void onMessage(Collection logSequences) {
  System.out.println("ExtendedJMSObjectGridEventListener[" +
    objectGridType + "].onMessage(): ");

  Iterator iter = logSequences.iterator();

  while (iter.hasNext()) {
            LogSequence seq = (LogSequence) iter.next();

      StringBuffer buffer = new StringBuffer();
      String mapName = seq.getMapName();
      int size = seq.size();
      buffer.append("\nLogSequence[mapName=" + mapName + ", size=" + size + ",
     objectGridType=" + objectGridType
        + "]: ");

      Iterator logElementIter = seq.getAllChanges();
      for (int i = seq.size() - 1; i >= 0; --i) {
       LogElement le = (LogElement) logElementIter.next();
       buffer.append(le.getType() + " -> key=" + le.getCacheEntry().getKey() + ", ");
      }
      buffer.append("\n");

      receivedLogSequenceList.add(buffer.toString());

    if (debug) {
       System.out.println("ExtendedJMSObjectGridEventListener["
      + objectGridType + "].onMessage(): " + buffer.toString());
      }
  }
 }

 public String dumpReceivedLogSequenceList() {
  String result = "";
  int size = receivedLogSequenceList.size();
  result = result + "\nExtendedJMSObjectGridEventListener[" + objectGridType
    + "]: receivedLogSequenceList size = " + size + "\n";
for (int i = 0; i < size; i++) {
   result = result + receivedLogSequenceList.get(i) + "\n";
  }
  return result;
 }

 public String toString() {
  return "ExtendedJMSObjectGridEventListener["
    + objectGridType + " - " + this.grid + "]";
 }
}
```

Konfiguration

Die erweiterte JMSObjectGridEventListener-Klasse muss für den Mechanismus für die Clientinaktivierung und für den Mechanismus für die Peer-to-Peer-Replikation gleich konfiguriert werden. Im Folgenden sehen Sie ein Beispiel für die XML-Konfiguration:

```
<objectGrid name="PRICEGRID">
   <bean id="ObjectGridEventListener"
    className="com.ibm.websphere.samples.objectgrid.jms.
      price.ExtendedJMSObjectGridEventListener">
    <property name="invalidationModel" type="java.lang.String"
     value="CLIENT_SERVER_MODEL" description="" />
    <property name="invalidationStrategy" type="java.lang.String"
     value="INVALIDATE" description="" />
    <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String"
     value="jms/TCF" description="" />
    <property name="jms_topicJndiName" type="java.lang.String"
     value="GRID.PRICEGRID" description="" />
    <property name="jms_topicName" type="java.lang.String"
     value="GRID.PRICEGRID" description="" />
    <property name="jms_userid" type="java.lang.String" value=""
     description="" />
    <property name="jms_password" type="java.lang.String" value=""
     description="" />
   </bean>
   <backingMap name="PRICE" pluginCollectionRef="PRICE"></backingMap>
       </objectGrid>
```

Anmerkung: Der Klassenname der Bean "ObjectGridEventListener" wird mit der erweiterten JMSObjectGridEventListener-Klasse mit denselben Eigenschaften wie die generische JMSObjectGridEventListener-Klasse konfiguriert.

ObjectGrid-XML-Deskriptordatei

Verwenden Sie zum Konfigurieren von WebSphere eXtreme Scale eine ObjectGrid-XML-Deskriptordatei und die API "ObjectGrid".

In den folgenden Abschnitten werden XML-Musterdateien bereitgestellt, um verschiedene Konfigurationen zu veranschaulichen. Jedes Element und Attribut der XML-Datei wird definiert. Verwenden Sie das ObjectGrid-XML-Deskriptorschema, um die XML-Deskriptordatei zu erstellen. Ein Beispiel für die ObjectGrid-XML-Deskriptordatei finden Sie im Abschnitt „Datei objectGrid.xsd" auf Seite 162.

Es wird eine geänderte Version der ursprünglichen Datei `companyGrid.xml` verwendet. Die folgende Datei `companyGridSingleMap.xml` gleicht der Datei `companyGrid.xml`. Die Datei `companyGridSingleMap.xml` hat eine einzige Map, und die Datei `companyGrid.xml` hat vier Maps. Die Elemente und Attribute der Datei sind im Anschluss an das Beispiel detailliert beschrieben.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

  <objectGrids>
<objectGrid name="CompanyGrid">
      <backingMap name="Customer"/>
    </objectGrid>
  </objectGrids>
</objectGridConfig>
```

Element "objectGridConfig"

Das Element "objectGridConfig" ist das Ausgangselement der XML-Datei. Schreiben Sie dieses Element in Ihrem eXtreme-Scale-XML-Dokument, wie im vorherigen Beispiel gezeigt. Dieses Element konfiguriert den Namespace der Datei und die Schemaposition. Das Schema ist in der Datei `objectGrid.xsd` definiert.

- Anzahl der Vorkommen: 1
- Untergeordnete Elemente: objectGrids und backingMapPluginCollections

Element "objectGrids"

Das Element "objectGrids" ist ein Container für alle objectGrid-Elemente. In der Datei `companyGridSingleMap.xml` enthält das Element "objectGrids" ein einziges ObjectGrid mit dem Namen "CompanyGrid".

- Anzahl der Vorkommen: 1 oder mehr
- Untergeordnetes Element: objectGrid

Element "objectGrid"

Verwenden Sie das Element "objectGrid", um ein ObjectGrid zu definieren. Jedes der Attribute im Element "objectGrid" entspricht einer Methode in der Schnittstelle "ObjectGrid".

- Anzahl der Vorkommen: 1 bis viele
- Untergeordnete Elemente: bean, backingMap, querySchema und streamQuerySet

Attribute

name
Gibt den Namen an, der dem ObjectGrid zugeordnet wird. Die XML-Validierung schlägt fehl, wenn dieses Attribut fehlt. (Erforderlich)

securityEnabled
Wenn Sie dieses Attribut auf `true` setzen, wird die Sicherheit auf ObjectGrid-Ebene aktiviert, woraufhin die Zugriffberechtigungen für die Daten in der Map aktiviert werden. Der Standardwert ist `true`. (Optional)

authorizationMechanism
Legt den Berechtigungsmechanismus für das Element fest. Sie können das Attribut auf einen der folgenden Werte setzen: `AUTHORIZATION_MECHANISM_JAAS` oder `AUTHORIZATION_MECHANISM_CUSTOM`. Der Standardwert ist `AUTHORIZATION_MECHANISM_JAAS`. Setzen Sie das Attribut auf `AUTHORIZATION_MECHANISM_CUSTOM`, wenn Sie ein angepasstes MapAuthorization-Plug-in verwenden. Sie müssen das Attribut "securityEnabled" auf `true` setzen, damit das Attribut "authorizationMechanism" wirksam wird. (Optional)

permissionCheckPeriod
Gibt einen ganzzahligen Wert in Sekunden an, der angibt, wie oft die Berechtigung geprüft wird, die verwendet wird, um einen Clientzugriff zuzulassen. Der Standardwert ist `0`. Wenn Sie das Attribut auf `0` setzen, wird der Berechtigungsmechanismus (Java Authentication and Authorization Service (JAAS) oder angepasste Berechtigung) bei jedem Aufruf der Methoden "get", "put", "update", "remove" und "evict" aufgefordert, zu prüfen, ob das aktuelle Subject-Objekt berechtigt ist. Ein Wert größer als 0 gibt die Anzahl an Sekunden an, für die eine Gruppe von Berechtigungen zwischengespeichert wird, bevor sie zur Aktualisierung an den Berechtigungsmechanismus zurückgegeben wird. Sie

müssen das Attribut "securityEnabled" auf `true` setzen, damit das Attribut "permissionCheckPeriod" wirksam wird. (Optional)

txTimeout
Gibt die Zeit in Sekunden an, die einer Transaktion für die Ausführung zugestanden wird. Wenn eine Transaktion nicht innerhalb dieser Zeit abgeschlossen wird, wird sie für Rollback markiert, und es wird eine Ausnahme des Typs "TransactionTimeoutException" ausgelöst. Wenn Sie das Attribut auf den Wert `0` setzen, existiert kein Zeitlimit für die Transaktionsausführung. (Optional)

entityMetadataXMLFile
Gibt den relativen Pfad zur XML-Deskriptordatei der Entität an. Der Pfad ist relativ zur Position der Objectgrid-Deskriptordatei. Verwenden Sie dieses Attribut, um ein Entitätsschema über eine XML-Datei zu definieren. Entitäten müssen vor dem Starten von eXtreme Scale definiert werden, so dass jede Entität an eine BackingMap gebunden werden kann. (Optional)

```
<objectGrid
(1) name="objectGridName"
(2) securityEnabled="true" | "false"
(3) authorizationMechanism="AUTHORIZATION_MECHANISM_JASS" | "AUTHORIZATION_MECHANISM_CUSTOM"
(4)  permissionCheckPeriod="permission_check_period"
(5) txTimeout="seconds"
(6) entityMetadataXMLFile="URL"
/>
```

Im folgenden Beispiel demonstriert die Datei `companyGridObjectGridAttr.xml` eine Methode zum Konfigurieren der Attribute eines Element "objectGrid". Die Sicherheit wird aktiviert, der Berechtigungsmechanismus wird auf "JAAS" gesetzt, und das Intervall für die Berechtigungsprüfung wird auf 45 Sekunden gesetzt. Außerdem registriert die Datei Entitäten, weil ein Attribut "entityMetadataXMLFile" angegeben wird.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlnc:xsi="http:www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

 <objectGrids>
  <objectGrid name="CompanyGrid" securityEnabled="true"
   authorizationMechanism="AUTHORIZATION_MECHANISM_JASS"
   permissionCheckPeriod="45"
   entityMetadataXMLFile="companyGridEntities.xml">
   <backingMap name="Customer"/>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridObjectGridAttr.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);

companyGrid.setSecurityEnabled();
companyGrid.setAuthorizationMechanism(SecurityConstants.AUTHORIZATION_MECHANISM_JAAS);
companyGrid.setPermissionCheckPeriod(45);
companyGrid.registerEntities(new URL("file:companyGridEntities.xml"));
```

Element "backingMap"

Das Element "backingMap" wird verwendet, um eine BackingMap-Instanz in einem ObjectGrid zu definieren. Jedes der Attribute im Element "backingMap" entspricht einer Methode in der Schnittstelle "BackingMap". Einzelheiten hierzu finden Sie in den Informationen zur Schnittstelle BackingMap im *Programmierhandbuch*.

- Anzahl der Vorkommen: 0 bis viele

- Untergeordnetes Element: timeBasedDBUpdate

Attribute

name
Gibt den Namen an, der der backingMap-Instanz zugeordnet wird. Die XML-Validierung schlägt fehl, wenn dieses Attribut fehlt. (Erforderlich)

readOnly
Definiert eine BackingMap-Instanz mit Lese-/Schreibzugriff, wenn Sie den Wert `false` für das Attribut angeben. Wenn Sie den Wert `true` für das Attribut angeben, ist die BackingMap-Instanz schreibgeschützt. Aufrufe der Methode "Session.getMap(String)" führen zur Erstellung einer dynamischen Map, wenn der an die Methode übergebene Name dem regulären Ausdruck entspricht, der im Namensattribut dieser BackingMap angegeben ist. Der Standardwert ist `false`. (Optional)

template
Gibt an, ob dynamische Maps verwendet werden können. Setzen Sie dieses Attribut auf `true`, wenn die BackingMap-Map eine Schablonen-Map ist. Schablonen-Maps können verwendet werden, um Maps nach dem Start von ObjectGrid dynamisch zu starten. Aufrufe der Methode "Session.getMap(String)" führen zur Erstellung dynamischer Maps, wenn der an die Methode übergebene Name dem regulären Ausdruck entspricht, der im Namensattribut der BackingMap angegeben ist. Der Standardwert ist `false`. (Optional)

pluginCollectionRef
Gibt eine Referenz auf ein backingMapPluginCollection-Plug-in an. Der Wert dieses Attributs muss mit dem Attribut "ID" eines backingMapCollection-Plug-ins übereinstimmen. Die Validierung schlägt fehl, wenn keine übereinstimmende ID vorhanden ist. Setzen Sie das Attribut, um BackingMap-Plug-ins wiederzuverwenden. (Optional)

numberOfBuckets
Gibt die Anzahl der für die BackingMap-Instanz zu verwendenden Buckets an. Die BackingMap-Instanz verwendet eine Hash-Tabelle für ihre Implementierung. Wenn mehrere Einträge in der BackingMap vorhanden sind, kann durch mehr Buckets eine bessere Leistung erzielt werden, weil das Risiko von Kollisionen mit zunehmender Anzahl an Buckets sinkt. Werden zusätzliche Buckets verwendet, sind auch mehr gemeinsame Zugriffe möglich. Geben Sie den Wert `0` an, um den nahen Cache in einem Client zu inaktivieren, wenn die Kommunikation mit eXtreme Scale über Fernzugriff erfolgt. Wenn Sie das Attribut für einen Client auf `0` setzen, definieren Sie den Wert nur in der ObjectGrid-XML-Deskriptordatei für Clientkorrekturwerte. (Optional)

preloadMode
Legt den Modus für vorheriges Laden (Preload) fest, wenn ein Loader-Plug-in für diese BackingMap-Instanz definiert ist. Der Standardwert ist `false`. Wenn Sie das Attribut auf `true` setzen, wird die Methode "Loader.preloadMap(Session, BackingMap)" asynchron aufgerufen. Andernfalls wird die Methode beim Laden von Daten blockiert, so dass der Cache bis zum Abschluss des Preload-Prozesses nicht verfügbar ist. Der Preload-Prozess findet während der Initialisierung statt. (Optional)

lockStrategy
Gibt an, ob der interne Sperrenmanager verwendet wird, wenn eine Transaktion auf einen Map-Eintrag zugreift. Setzen Sie dieses Attribut auf einen der folgenden drei Werte: `OPTIMISTIC`, `PESSIMISTIC` oder `NONE`. Der Standardwert ist `OPTIMISTIC`. (Optional)

Die optimistische Sperrstrategie wird gewöhnlich verwendet, wenn eine Map kein Loader-Plug-in hat, wenn hauptsächlich Leseoperationen und weniger Schreib- und Aktualisierungsoperationen in der Map durchgeführt werden und wenn das Sperren nicht vom Persistenzmanager, der eXtreme Scale als Nebencache verwendet, oder von der Anwendung übernommen wird. Eine exklusive Sperre wird für einen Map-Eintrag angefordert, der bei der Festschreibung eingefügt, aktualisiert oder entfernt wird. Die Sperre stellt sicher, dass die Versionsinformationen von anderen Transaktionen nicht geändert werden können, während die Transaktion, die festgeschrieben wird, eine optimistische Versionsprüfung durchführt.

Die pessimistische Sperrstrategie wird gewöhnlich für eine Map verwendet, die kein Loader-Plug-in hat, und wenn das Sperren nicht von einem Persistenzmanager, der eXtreme Scale als Nebencache verwendet, von einem Loader-Plug-in oder von der Anwendung übernommen wird. Die pessimistische Sperrstrategie wird verwendet, wenn bei der optimistischen Sperrstrategie zu häufig Fehler auftreten, weil Aktualisierungstransaktionen für denselben Map-Eintrag zu häufig kollidieren.

Die Strategie ohne Sperren zeigt an, dass der interne Sperrenmanager nicht benötigt wird. Die Steuerung des gemeinsamen Zugriffs erfolgt außerhalb von eXtreme Scale durch den Persistenzmanager, der eXtreme Scale als Nebencache verwendet, durch die Anwendung oder durch das Loader-Plug-in, das Datenbanksperren für die Steuerung des gemeinsamen Zugriffs verwendet.

Weitere Informationen finden Sie in den Informationen zum Sperren von Map-Einträgen im *Programmierhandbuch*.

numberOfLockBuckets
Legt die Anzahl der Sperr-Buckets fest, die vom Sperrenmanager für die BackingMap-Instanz verwendet werden. Setzen Sie das Attribut "lockStrategy" auf `OPTIMISTIC` oder `PESSIMISTIC`, um einen Sperrenmanager für die BackingMap-Instanz zu erstellen. Der Sperrenmanager verwendet eine Hash-Tabelle, um die Einträge zu verfolgen, die von einer oder mehreren Transaktionen gesperrt werden. Wenn viele Einträge vorhanden sind, kann durch mehr Buckets eine bessere Leistung erzielt werden, weil das Risiko von Kollisionen mit zunehmender Anzahl an Buckets sinkt. Werden zusätzliche Buckets verwendet, sind auch mehr gemeinsame Zugriffe möglich. Setzen Sie das Attribut "lockStrategy" auf `NONE`, um anzugeben, dass die BackingMap-Instanz keinen Sperrenmanager verwendet. (Optional)

lockTimeout
Legt das Zeitlimit für Sperre fest, das vom Sperrenmanager für die BackingMap-Instanz verwendet wird. Setzen Sie das Attribut "lockStrategy" auf `OPTIMISTIC` oder `PESSIMISTIC`, um einen Sperrenmanager für die BackingMap-Instanz zu erstellen. Zur Vermeidung von Deadlocks hat der Sperrenmanager ein Standardzeitlimit von 15 Sekunden. Bei Überschreitung des Zeitlimits wird eine Ausnahme des Typs "LockTimeoutException" ausgelöst. Der Standardwert von 15 Sekunden ist für die meisten Anwendungen ausreichend, aber auf einem System mit hoher Belastung können Zeitlimitüberschreitungen auch dann auftreten, wenn kein Deadlock vorhanden ist. Verwenden Sie das Attribut "lockTimeout", um einen höheren als den Standardwert festzulegen, um falsche Zeitlimitüberschreitungsausnahmen zu verhindern. Setzen Sie das Attribut "lockStrategy" auf `NONE`, um anzugeben, dass die BackingMap-Instanz keinen Sperrenmanager verwendet. (Optional)

CopyMode
Gibt an, ob eine get-Operation eines Eintrags in der BackingMap-Instanz den

tatsächlichen Wert, eine Kopie des Werts oder einen Proxy für den Wert zurückgibt. Setzen Sie das Attribut "CopyMode" auf einen der folgenden fünf Werte:

COPY_ON_READ_AND_COMMIT
Der Standardwert ist COPY_ON_READ_AND_COMMIT. Setzen Sie das Attribut auf COPY_ON_READ_AND_COMMIT, um sicherzustellen, dass eine Anwendung keine Referenz auf das Wertobjekt verwendet, das in der BackingMap-Instanz enthalten ist. Stattdessen arbeitet die Anwendung immer mit einer Kopie des Werts, der in der BackingMap-Instanz enthalten ist. (Optional)

COPY_ON_READ
Setzen Sie das Attribut auf COPY_ON_READ, um eine im Vergleich mit dem Wert COPY_ON_READ_AND_COMMIT höhere Leistung zu erzielen, indem der Kopiervorgang bei der Festschreibung einer Transaktion umgangen wird. Zur Gewährleistung der Integrität der BackingMap-Daten stellt die Anwendung sicher, dass jede Referenz auf einen Eintrag nach der Festschreibung der Transaktion gelöscht wird. Durch die Festlegung dieses Werts wird eine Methode "ObjectMap.get" aufgerufen, die eine Kopie des Werts an Stelle einer Referenz auf den Wert zurückgibt, was sicherstellt, dass Änderungen, die von der Anwendung am Wert vorgenommen werden, erst dann für das BackingMap-Element wirksam werden, wenn die Transaktion festgeschrieben wird.

COPY_ON_WRITE
Setzen Sie das Attribut auf COPY_ON_WRITE, um eine im Vergleich mit dem Wert COPY_ON_READ_AND_COMMIT bessere Leistung zu erzielen, in dem der Kopiervorgang beim ersten Aufruf der Methode "ObjectMap.get" für einen bestimmten Schlüssel durch eine Transaktion umgangen wird. Stattdessen gibt die Methode "ObjectMap.get" einen Proxy für den Wert an Stelle einer Referenz auf das Wertobjekt zurück. Der Proxy stellt sicher, dass keine Kopie des Werts erstellt wird, sofern die Anwendung nicht eine Methode "set" für die Wertschnittstelle aufruft.

NO_COPY
Setzen Sie das Attribut auf den Wert NO_COPY, um einer Anwendung zu ermöglichen, ein Wertobjekt, das mit einer Methode "ObjectMap.get" abgerufen wird, nie zu ändern, um so Leistungsverbesserungen zu erzielen. Setzen Sie das Attribut für Maps, die Entitäten der EntityManager-API zugeordnet sind, auf NO_COPY.

COPY_TO_BYTES
Setzen Sie das Attribut auf COPY_TO_BYTES, um den Speicherbedarf für komplexe Objekttypen zu verringern und die Leistung zu verbessern, wenn das Kopieren des Objekts durch Serialisierung vorgenommen werden soll. Wenn ein Objekt nicht klonbar ist oder kein angepasstes ObjectTransformer-Plug-in mit einer effizienten Methode "copyValue" bereitgestellt wird, wird der Standardkopiermechanismus verwendet, um das Objekt zu serialisieren und zu dekomprimieren und die Kopie zu erstellen. Bei der Einstellung COPY_TO_BYTES wird die Dekomprimierung bei einer reinen Leseoperation und die Serialisierung bei einer reinen Festschreibungsoperation durchgeführt.

Weitere Informationen zu diesen Einstellungen finden Sie in den Informationen zu den bewährten Verfahren für CopyMode im *Programmierhandbuch*..

valueInterfaceClassName
Gibt eine Klasse an, die erforderlich ist, wenn Sie das Attribut "CopyMode" auf

COPY_ON_WRITE setzen. Dieses Attribut wird für alle anderen Modi ignoriert. Beim Wert COPY_ON_WRITE wird ein Proxy verwendet, wenn die Methode "ObjectMap.get" aufgerufen wird. Der Proxy stellt sicher, dass keine Kopie des Werts erstellt wird, sofern die Anwendung keine set-Methode in der Klasse aufruft, die mit dem Attribut "valueInterfaceClassName" angegeben ist. (Optional)

copyKey
Gibt an, ob eine Kopie des Schlüssels erforderlich ist, wenn ein Map-Eintrag erstellt wird. Wenn das Schlüsselobjekt kopiert wird, kann die Anwendung dasselbe Schlüsselobjekt für jede ObjectMap-Operation verwenden. Setzen Sie das Attribut auf true, um das Schlüsselobjekt zu kopieren, wenn ein Map-Eintrag erstellt wird. Der Standardwert ist false. (Optional)

nullValuesSupported
Setzen Sie das Attribut auf true, um Nullwerte in der ObjectMap zu unterstützen. Wenn Nullwerte unterstützt werden, bedeutet eine get-Operation, die null zurückgibt, dass der Wert null ist oder dass die Map den an die Methode übergebenen Schlüssel nicht enthält. Der Standardwert ist true. (Optional)

ttlEvictorType
Gibt an, wie die Verfallszeit eines BackingMap-Eintrags berechnet wird. Setzen Sie dieses Attribut auf einen der folgenden Werte: CREATION_TIME, LAST_ACCESS_TIME, LAST_UPDATE_TIME oder NONE. Der Wert CREATION_TIME zeigt an, dass die Verfallszeit eines Eintrags die Summe aus Erstellungszeit des Eintrags und Wert des Attributs "timeToLive" ist. Der Wert LAST_ACCESS_TIME zeigt an, dass die Verfallszeit eines Eintrags die Summe aus letzter Zugriffszeit des Eintrags (unabhängig davon, ob der Eintrag aktualisiert oder nur gelesen wurde) und Wert des Attributs "timeToLive" ist. Der Wert LAST_UPDATE_TIME zeigt an, dass die Verfallszeit eines Eintrags die Summe aus letzter Aktualisierungszeit des Eintrags und Wert des Attributs "timeToLive" ist. Der Wert NONE (der Standardwert) zeigt an, dass ein Eintrag keine Verfallszeit hat und so lange in der BackingMap-Instanz bleibt, bis die Anwendung den Eintrag explizit entfernt oder ungültig macht. (Optional)

timeToLive
Gibt an, wie lange (in Sekunden) jeder Map-Eintrag existiert. Der Standardwert 0 bedeutet, dass der Map-Eintrag unbegrenzt bzw. so lange existiert, bis die Anwendung den Eintrag explizit entfernt oder ungültig macht. Andernfalls entfernt der TTL-Evictor den Map-Eintrag auf der Basis dieses Werts. (Optional)

streamRef
Gibt an, dass die BackingMap die Quellen-Map eines Datenstroms ist. Alle Einfüge- oder Aktualisierungsoperationen für die BackingMap werden in ein Streaming-Ereignis an die Abfragesteuerkomponente für Datenströme konvertiert. Dieses Attribut muss auf einen gültigen Datenstromnamen in einem streamQuerySet-Objekt verweisen. (Optional)

viewRef
Gibt an, dass die BackingMap eine Sicht-Map ist. Die Sichtausgabe der Abfragesteuerkomponente für Datenströme wird in das eXtreme-Scale-Tupelformat konvertiert und in der Map gespeichert. (Optional)

writeBehind
Gibt an, dass die Write-behind-Unterstützung mit Write-behind-Parametern aktiviert ist. (Optional). Write-behind-Parameter setzen sich aus einer maximalen Aktualisierungszeit und einer maximalen Anzahl an Schlüsselaktualisierungen

zusammen. Das Format des Write-behind-Parameters ist "[T(Zeit)][;][C(Zähler)]". Die Datenbank wird aktualisiert, wenn eines der folgenden Ereignisse eintritt:

- Die maximale Aktualisierungszeit in Sekunden seit der letzten Aktualisierung ist abgelaufen.
- Die Anzahl verfügbarer Aktualisierungen in der Warteschlangen-Map hat die maximal zulässige Aktualisierungsanzahl erreicht.

Weitere Informationen finden Sie unter „Unterstützung des Write-behind-Cachings” auf Seite 128.

Die Write-behind-Unterstützung ist eine Erweiterung des Loader-Plug-ins, das Sie verwenden, um eXtreme Scale mit der Datenbank zu integrieren. Sehen Sie sich beispielsweise die Informationen zur Konfiguration eines JPA-Loaders im Abschnitt „JPA-Loader konfigurieren” auf Seite 235 an.

evictionTriggers
Legt die Typen zusätzlich zu verwendender Bereinigungs-Trigger fest. Alle Evictor (Bereinigungsprogramme) für die BackingMap verwenden diese liste zusätzlicher Trigger. Zur Vermeidung von Ausnahmen des Typs "IllegalStateException" muss dieses Attribut vor der Methode "ObjectGrid.initialize()" aufgerufen werden. Beachten Sie auch, dass die Methode "ObjectGrid.getSession()" die Methode "ObjectGrid.initialize()" implizit aufruft, falls die Methode noch nicht von der Anwendung aufgerufen wurde. Einträge in der Trigger-Liste werden durch Semikolons getrennt. Zu den aktuellen Bereinigungs-Triggern gehört MEMORY_USAGE_THRESHOLD. (Optional)

```
<backingMap
(1)   name="objectGridName"
(2)   readOnly="true" | "false"
(3)      template="true" | "false"
(4)   pluginCollectionRef="reference to backingMapPluginCollection"
(5)   numberOfBuckets="number of buckets"
(6)   preloadMode="true" | "false"
(7)   lockStrategy="OPTIMISTIC" | "PESSIMISTIC" | "NONE"
(8)   numberOfLockBuckets="number of lock buckets"
(9)   lockTimeout="lock timeout"
(10)   copyMode="COPY_ON_READ_AND_COMMIT" | "COPY_ON_READ" | "COPY_ON_WRITE"
       | "NO_COPY" | "COPY_TO_BYTES"
(11)   valueInterfaceClassName="value interface class name"
(12)   copyKey="true" | "false"
(13)   nullValuesSupported="true" | "false"
(14)   ttlEvictorType="CREATION_TIME" | "LAST_ACCESS_TIME" | "LAST_UPDATE_TIME" | NONE"
(15)   timeToLive="time to live"
(16)   streamRef="reference to a stream"
(17)   viewRef="reference to a view"
(18)   writeBehind="write-behind parameters"
(19)   evictionTriggers="MEMORY_USAGE_THRESHOLD"
/>
```

Im folgenden Beispiel wird die Datei companyGridBackingMapAttr.xml verwendet, um eine BackingMap-Beispielkonfiguration zu veranschaulichen.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

 <objectGrids>
<objectGrid name="CompanyGrid">
    <backingMap name="Customer" readOnly="true"
     numberOfBuckets="641" preloadMode="false"
     lockStrategy="OPTIMISTIC" numberOfLockBuckets="409"
     lockTimeout="30" copyMode="COPY_ON_WRITE"
     valueInterfaceClassName="com.ibm.websphere.samples.objectgrid.CounterValueInterface"
     copyKey="true" nullValuesSupported="false"
     ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3000"/>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Der folgende Mustercode demonstriert den programmgesteuerten Ansatz, um dieselbe Konfiguration zu erhalten wie mit der Datei `companyGridBackingMapAttr.xml` aus dem vorherigen Beispiel:

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);

BackingMap customerMap = companyGrid.defineMap("Customer");
customerMap.setReadOnly(true);
customerMap.setNumberOfBuckets(641);
customerMap.setPreloadMode(false);
customerMap.setLockStrategy(LockStrategy.OPTIMISTIC);
customerMap.setNumberOfLockBuckets(409);
customerMap.setLockTimeout(30);

// Wenn der Kopiermodus auf COPY_ON_WRITE gesetzt wird, ist eine Klasse "valueInterface" erforderlich.
customerMap.setCopyMode(CopyMode.COPY_ON_WRITE,
 com.ibm.websphere.samples.objectgrid.CounterValueInterface.class);
customerMap.setCopyKey(true);
customerMap.setNullValuesSupported(false);
customerMap.setTtlEvictorType(TTLType.LAST_ACCESS_TIME);
customerMap.setTimeToLive(3000); // Lebensdauer (TTL) auf 50 Minuten setzen
```

Element "bean"

Verwenden Sie das Element "bean", um Plug-ins zu definieren. Sie können Plug-ins objectGrid- und BackingMap-Elementen zuordnen.

- Anzahl der Vorkommen im Element "objectGrid": 0 bis viele
- Anzahl der Vorkommen im Element "backingMapPluginCollection": 0 bis viele
- Untergeordnetes Element: property

Attribute

id Gibt den Typ des zu erstellenden Plug-ins an. (Erforderlich)

Die gültigen Plug-ins für eine Bean, die ein untergeordnetes Element des Elements "objectGrid" ist, sind in der folgenden Liste aufgeführt:

- TransactionCallback-Plug-in
- ObjectGridEventListener-Plug-in
- SubjectSource-Plug-in
- MapAuthorization-Plug-in
- SubjectValidation-Plug-in

Die gültigen Plug-ins für eine Bean, die ein untergeordnetes Element des Elements "backingMapPluginCollection" ist, sind in der folgenden Liste aufgeführt:

- Loader-Plug-in
- ObjectTransformer-Plug-in
- OptimisticCallback-Plug-in
- Evictor-Plug-in
- MapEventListener-Plug-in
- MapIndex-Plug-in

className
Gibt den Namen der Klasse bzw. SpringBean an, die zum Erstellen des Plug-ins instanziiert werden soll. Die Klasse muss die Schnittstelle für den Plug-in-Typ implementieren. Wenn Sie beispielsweises `ObjectGridEventListener` als Wert für das Attribut "id" angeben, muss der Wert des Attributs "className" auf eine Klasse verweisen, die die Schnittstelle "ObjectGridEventListener" implementiert. (Erforderlich)

```
<bean
(1) id="TransactionCallback" | "ObjectGridEventListener" |"SubjectSource" |
    "MapAuthorization" | "SubjectValidation" | "Loader" | "ObjectTransformer" |
    "OptimisticCallback" | "Evictor" | "MapEventListener" | "MapIndexPlugin"
(2) className="class name" | "(spring)bean name"
/>
```

Im folgenden Beispiel wird die Datei companyGridBean.xml verwendet, um zu veranschaulichen, wie Plug-ins mit dem Element "bean" konfiguriert werden. Ein ObjectGridEventListener-Plug-in wird dem ObjectGrid "CompanyGrid" hinzugefügt. Das Attribut "className" für dieses Element "bean" ist "com.ibm.websphere.objectgrid.plugins.builtins.TranPropListener". Diese Klasse implementiert die Schnittstelle "com.ibm.websphere.objectgrid.plugins.ObjectGridEventListener".

Es wird auch ein BackingMap-Plug-in in der Datei companyGridBean.xml definiert. Ein Evictor-Plug-in wird der BackingMap-Instanz "Customer" hinzugefügt. Das die Bean-ID "Evictor" lautet, muss das Attribut "className" eine Klasse angeben, die die Schnittstelle "com.ibm.websphere.objectgrid.plugins.Evictor" implementiert. Die Klasse "com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" implementiert diese Schnittstelle. Die BackingMap referenziert ihre Plug-ins über das Attribut "pluginCollectionRef".

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

 <objectGrids>
<objectGrid name="CompanyGrid">
   bean id="ObjectGridEventListener"
    className="com.ibm.websphere.objectgrid.plugins.builtins.TranPropListener"/>
   <backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
<backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor/>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei companyGridBean.xml aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
TranPropListener tranPropListener = new TranPropListener();
companyGrid.addEventListener(tranPropListener);

BackingMap customerMap = companyGrid.defineMap("Customer");
Evictor lruEvictor = new com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor();
customerMap.setEvictor(lruEvictor);
```

Element "property"

Verwenden Sie das Element "property", um Plug-ins Eigenschaften hinzuzufügen. Der Name der Eigenschaft muss einer set-Methode in der von der übergeordneten Bean referenzierten Klasse entsprechen.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

name
Der Name der Eigenschaft. Der Wert, der diesem Attribut zugeordnet wird, muss einer set-Methode in der Klasse entsprechen, die mit dem Attribut "className" der übergeordneten Bean angegeben wird. Wenn Sie beispielsweise das Attribut "className" der Bean auf `com.ibm.MyPlugin` setzen und der angegebene Name der Eigenschaft `size` lautet, muss die Klasse `com.ibm.MyPlugin` eine Methode "setSize" enthalten. (Erforderlich)

type
Gibt den Typ der Eigenschaft an. Der Typ wird an die set-Methode übergeben, die mit dem Attribut "name" angegeben wurde. Die gültigen Werte sind primitive Java-Typen, ihre java.lang-Pendants und java.lang.String. Die Attribute "name" und "type" müssen einer Methodensignatur im Attribut "className" der Bean entsprechen. Wenn Sie beispielsweise den Namen `size` und den Typ `int` festlegen, muss eine Methode "setSize(int)" in der Klasse vorhanden sein, die mit dem Attribut "className" für die Bean angegeben wurde. (Erforderlich)

value
Gibt den Wert der Eigenschaft an. Dieser Wert wird in den mit dem Attribut "type" angegebenen Typ konvertiert und anschließend als Parameter in dem Aufruf der set-Methode verwendet, die mit den Attributen "name" und "type" angegeben wurde. Der Wert dieses Attributs wird nicht validiert. (Erforderlich)

description
Beschreibt die Eigenschaft. (Optional)

```
<bean
(1)   name="name"
(2) type="java.lang.String" | "boolean" | "java.lang.Boolean" | "int" |
       "java.lang.Integer" | "double" | "java.lang.Double" | "byte" |
       "java.lang.Byte" | "short" | "java.lang.Short" | "long" |
       "java.lang.Long" | "float" | "java.lang.Float" | "char" |
       "java.lang.Character"
(3)   value="value"
(4) description="description"
/>
```

Im folgenden Beispiel wird die Datei `companyGridProperty.xml` verwendet, um zu veranschaulichen, wie einer Bean ein Element "property" hinzugefügt wird. In diesem Beispiel wird einem Evictor eine Eigenschaft mit dem Namen "maxSize" und dem Typ "int" hinzugefügt. Der Evictor "com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" hat eine Methodensignatur, die der Methode "setMaxSize(int)" entspricht. Der ganzzahlige Wert 499 wird an die Methode "setMaxSize(int)" in der Klasse "com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" übergeben.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

 <objectGrids>
<objectGrid name="CompanyGrid">
   <backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
<backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor>
     <property name="maxSize" type="int" value="449"
       description="The maximum size of the LRU Evictor"/>
   </bean>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei companyGridProperty.xml aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);

BackingMap customerMap = companyGrid.defineMap("Customer");

LRUEvictor lruEvictor = new com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor();
// Wenn Sie die XML-Datei verwenden würden, würde die hinzugefügte
// Eigenschaft den folgenden Aufruf bewirken
lruEvictor.setMaxSize(449);
customerMap.setEvictor(lruEvictor);
```

Element "backingMapPluginsCollections"

Das Element "backingMapPluginsCollections" ist ein Container für alle backingMapPluginCollection-Elemente. In der Datei companyGridProperty.xml aus dem vorherigen Beispiel enthält das Element "backingMapPluginCollections" ein einziges Element "backingMapPluginCollection" mit der ID customerPlugins.

- Anzahl der Vorkommen: 0 bis 1
- Untergeordnetes Element: backingMapPluginCollection

Element "backingMapPluginCollection"

Das Element "backingMapPluginCollection" definiert die BackingMap-Plug-ins und wird über das Attribut **id** identifiziert. Geben Sie das Attribut "pluginCollectionRef" an, um die Plug-ins zu referenzieren. Wenn Sie mehrere BackingMap-Plug-ins auf ähnliche Weise konfigurieren, kann jede BackingMap dasselbe Element "backingMapPluginCollection" referenzieren.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: bean

Attribute

id Identifiziert die backingMapPluginCollection und wird vom Attribut "pluginCollectionRef" des Elements "backingMap" referenziert. Jede ID muss eindeutig sein. Wenn der Wert des Attributs "pluginCollectionRef" nicht mit der ID eines der backingMapPluginCollection-Elemente übereinstimmt, schlägt die XML-Validierung fehl. Es können beliebig viele backingMap-Elemente ein einziges Element "backingMapPluginCollection" referenzieren. (Erforderlich)

```
<backingMapPluginCollection
(1)   id="id"
/>
```

Im folgenden Beispiel wird die Datei companyGridCollection.xml verwendet, um zu veranschaulichen, wie das Element "backingMapPluginCollection" verwendet wird. In dieser Datei verwendet die BackingMap "Customer" die backingMapPluginCollection "customerPlugins", um die BackingMap "Customer" mit einem LRUEvictor-Plug-in zu konfigurieren. Die BackingMaps "Item" und "OrderLine" referenzieren die backingMapPluginCollection "collection2". Für diese BackingMaps ist jeweils ein LFUEvictor-Plug-in definiert.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

 <objectGrids>
<objectGrid name="CompanyGrid">
   <backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
```

```
   <backingMap name="Item" pluginCollectionRef="collection2"/>
   <backingMap name="OrderLine"
    pluginCollectionRef="collection2"/>
<backingMap name="Order"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
<backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor/>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="collection2">
   <bean id="Evictor"
     className="com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor/>
   <bean id="OptimisticCallback"
     className="com.ibm.websphere.samples.objectgrid.EmployeeOptimisticCallBackImpl"/>
  </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridCollection.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();

ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
BackingMap customerMap = companyGrid .defineMap("Customer");
LRUEvictor customerEvictor = new LRUEvictor();
customerMap.setEvictor(customerEvictor);

BackingMap itemMap = companyGrid.defineMap("Item");
LFUEvictor itemEvictor = new LFUEvictor();
itemMap.setEvictor(itemEvictor);

BackingMap orderLineMap = companyGrid.defineMap("OrderLine");
LFUEvictor orderLineEvictor = new LFUEvictor();
orderLineMap.setEvictor(orderLineEvictor);

BackingMap orderMap = companyGrid.defineMap("Order");
```

Element "querySchema"

Das Element "querySchema" definiert Beziehungen zwischen BackingMaps und identifiziert den Objekttyp jeder Map. Diese Informationen werden von ObjectQuery verwendet, um Zeichenfolgen in der Abfragesprache in Map-Zugriffsaufrufe zu übersetzen.

- Anzahl der Vorkommen: 0 bis 1
- Untergeordnete Elemente: mapSchemas, relationships

Element "mapSchemas"

Jedes Element "querySchema" hat ein einziges Element "mapSchemas", das ein oder mehrere mapSchema-Elemente enthält.

- Anzahl der Vorkommen: 1
- Untergeordnetes Element: mapSchema

Element "mapSchema"

Ein Element "mapSchema" definiert den Typ der Objekte, die in einer BackingMap gespeichert werden, und enthält Anweisungen zum Zugriff auf die Daten.

- Anzahl der Vorkommen: 1 oder mehr
- Untergeordnetes Element: Ohne

Attribute

mapName
Gibt den Namen der BackingMap an, die dem Schema hinzugefügt werden soll. (Erforderlich)

valueClass
Gibt den Typ des Objekts an, der im Wertabschnitt der BackingMap gespeichert wird. (Erforderlich)

primaryKeyField
Gibt den Namen des Primärschlüsselattributs im Attribut "valueClass" an. Der Primärschlüssel muss auch im Schlüsselabschnitt der BackingMap gespeichert werden. (Optional)

accessType
Gibt an, wie die Abfragesteuerkomponente die persistenten Daten in den valueClass-Objektinstanzen überwacht und auf diese zugreift. Wenn Sie das Attribut auf den Wert `FIELD` setzen, werden die Klassenfelder überwacht und dem Schema hinzugefügt. Wenn Sie das Attribut auf den Wert `PROPERTY` setzen, werden die Attribute, die den get- und is-Methoden zugeordnet sind, verwendet. Der Standardwert ist `PROPERTY`. (Optional)

```
<mapSchema
(1)   mapName="backingMapName"
(2)   valueClass="com.mycompany.OrderBean"
(3)   primaryKeyField="orderId"
(4)   accessType="PROPERTY" | "FIELD"
/>
```

Im folgenden Beispiel wird die Datei `companyGridQuerySchemaAttr.xml` verwendet, um eine Beispielkonfiguration für mapSchema zu veranschaulichen:

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

 <objectGrids>
<objectGrid name="CompanyGrid">
<backingMap name="Order"/>
   <backingMap name="Customer"/>

   <querySchema>
    <mapSchemas>
     <mapSchema mapName="Order"
      valueClass="com.mycompany.OrderBean"
      primaryKeyField="orderNumber"
      accessType="FIELD"/>
     <mapSchema mapName="Customer"
      valueClass="com.mycompany.CustomerBean"
      primaryKeyField="id"
      accessType="FIELD"/>
    </mapSchemas>
   </querySchema>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridQuerySchemaAttr.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Schema definieren
QueryConfig queryCfg = new QueryConfig();
queryCfg.addQueryMapping(new QueryMapping(
```

```
    "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryMapping(new QueryMapping(
    "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
companyGrid.setQueryConfig(queryCfg);
```

Element "relationships"

Jedes Element "querySchema" hat kein oder ein Element "relationships", das eine oder mehrere Elemente "relationship" enthält.

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: relationship

Element "relationship"

Ein Element "relationship" definiert die Beziehung zwischen zwei BackingMaps sowie die Attribute im Attribut "valueClass", die für die Beziehungsbindung verwendet werden.

- Anzahl der Vorkommen: 1 oder mehr
- Untergeordnetes Element: Ohne

Attribute

source
Gibt den Namen der valueClass der Quellenseite einer Beziehung an. (Erforderlich)

target
Gibt den Namen der valueClass der Zielseite einer Beziehung an. (Erforderlich)

relationField
Gibt den Namen des Attributs in der Quellen-valueClass an, die auf das Ziel verweist. (Erforderlich)

invRelationField
Gibt den Namen des Attributs in der Ziel-valueClass an, die auf die Quelle verweist. Wenn Sie dieses Attribut nicht angeben, ist die Beziehung unidirektional. (Optional)

```
<mapSchema
(1)   source="com.mycompany.OrderBean"
(2)   target="com.mycompany.CustomerBean"
(3)   relationField="customer"
(4)   invRelationField="orders"
/>
```

Im folgenden Beispiel wird die Datei `companyGridQuerySchemaWithRelationshipAttr.xml` verwendet, um eine mapSchema-Musterkonfiguration zu veranschaulichen, die eine bidirektionale Beziehung enthält.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

 <objectGrids>
<objectGrid name="CompanyGrid">
<backingMap name="Order"/>
   <backingMap name="Customer"/>

   <querySchema>
    <mapSchemas>
     <mapSchema mapName="Order"
      valueClass="com.mycompany.OrderBean"
      primaryKeyField="orderNumber"
      accessType="FIELD"/>
```

```
     <mapSchema mapName="Customer"
      valueClass="com.mycompany.CustomerBean"
      primaryKeyField="id"
      accessType="FIELD"/>
    </mapSchemas>
    <relationships>
     <relationship
      source="com.mycompany.OrderBean"
      target="com.mycompany.CustomerBean"
      relationField="customer"/>
      invRelationField="orders"/>
    </relationships>
   </querySchema>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridQuerySchemaWithRelationshipAttr.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Schema definieren
QueryConfig queryCfg = new QueryConfig();
queryCfg.addQueryMapping(new QueryMapping(
    "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryMapping(new QueryMapping(
    "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryRelationship(new QueryRelationship(
     OrderBean.class.getName(), CustomerBean.class.getName(), "customer", "orders"));
companyGrid.setQueryConfig(queryCfg);
```

Element "streamQuerySet"

Das Element "streamQuerySet" ist das Ausgangselement für die Definition eines Abfragesatzes für Datenströme.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnete Elemente: stream, view

Element "stream"

Das Element "stream" stellt einen Datenstrom für die Abfragesteuerkomponente für Datenströme dar. Jedes Attribut des Elements "stream" entspricht einer Methode in der Schnittstelle "StreamMetadata".

- Anzahl der Vorkommen: 1 bis viele
- Untergeordnetes Element: basic

Attribute

name
Gibt den Namen des Datenstroms an. Die Validierung schlägt fehl, wenn dieses Attribut nicht angegeben wird. (Erforderlich)

valueClass
Gibt den Klassentyp des Werts an, der in der ObjectMap des Datenstroms gespeichert wird. Der Klassentyp wird verwendet, um das Objekt in Datenstromereignisse zu konvertieren und um eine SQL-Anweisung zu generieren, wenn die Anweisung nicht angegeben ist. (Erforderlich)

sql
Gibt die SQL-Anweisung des Datenstroms an. Wenn Sie diese Eigenschaft nicht

angeben, wird eine Datenstrom-SQL durch Reflexion der Attribute bzw. Zugriffsmethoden im Attribut "valueClass" bzw. durch Verwendung der Tupelattribute der Entitätsmetadaten generiert. (Optional)

access
Gibt den Typ für den Zugriff auf die Attribute der Wertklasse an. Wenn Sie das Attribut auf den Wert `FIELD` setzen, werden die Attribute durch Java-Reflexion direkt aus den Feldern abgerufen. Andernfalls werden Zugriffsmethoden für das Lesen der Attribute verwendet. Der Standardwert ist `PROPERTY`. (Optional)

```
<stream
(1)   name="streamName"
(2)   valueClass="streamMapClassType"
(3)   sql="streamSQL create stream stockQuote
     keyed by t ( transactionvolume INTEGER, price DECIMAL (9,2), issue VARCHAR(100) );"
(4)   access="PROPERTY" | "FIELD"
/>
```

Element "view"

Das Element "view" stellt eine Datenstromabfragesicht dar. Jedes Element "stream" entspricht einer Methode in der Schnittstelle "ViewMetadata".

- Anzahl der Vorkommen: 1 bis viele
- Untergeordnete Elemente: basic, id

Attribute

name
Gibt den Namen der Sicht an. Die Validierung schlägt fehl, wenn dieses Attribut nicht angegeben wird. (Erforderlich)

sql
Gibt die SQL des Datenstroms an, die die Sichtkonvertierung definiert. Die Validierung schlägt fehl, wenn dieses Attribut nicht angegeben wird. (Erforderlich)

valueClass
Gibt den Klassentyp des Werts an, der in dieser Sicht der ObjectMap gespeichert wird. Der Klassentyp wird verwendet, um die Sichtereignisse in das richtige Tupelformat zu konvertieren, das mit diesem Klassentyp kompatibel ist. Wenn Sie den Klassentyp nicht angeben, wird ein Standardformat gemäß den Spaltendefinition in Stream Processing Technology Structured Query Language (SPTSQL) verwendet. Wenn Entitätsmetadaten für diese Sicht-Map definiert sind, verwenden Sie das Attribut "valueClass" nicht. (Optional)

access
Gibt den Typ für den Zugriff auf die Attribute der Wertklasse an. Wenn Sie den Zugriffstyp auf `FIELD` setzen, werden die Spaltenwerte durch Java-Reflexion direkt aus den Feldern abgerufen. Andernfalls werden Zugriffsmethoden für das Definieren der Attribute verwendet. Der Standardwert ist `PROPERTY`. (Optional)

```
<view
(1)   name="viewName"
(2)   valueClass="viewMapValueClass"
(3)   sql="viewSQL CREATE VIEW last5MinuteAvgPrice AS
      SELECT issue, avg(price) as totalVolume
        FROM (SELECT * FROM stockQuote FETCH LATEST 5 MINUTES) group by issue;"/>
(4)   access="PROPERTY" | "FIELD"
/>
```

Element "basic"

Das Element "basic" wird verwendet, um eine Zuordnung des Attributnamens in der Wertklasse bzw. den Entitätsmetadaten zu der in SPTSQL definierten Spalte zu definieren.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

```
<basic
(1)   name="attributeName"
(2)   column="columnName"
/>
```

Element "id"

Das Element "id" wird für die Zuordnung eines Schlüsselattributs verwendet.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

```
<id
(1)   name="idName"
(2)   column="columnName"
/>
```

Im folgenden Beispiel wird die Datei `StreamQueryApp2.xml` verwendet, um zu veranschaulichen, wie die Attribut eines Datenstromabfragesatzes konfiguriert werden. Der Datenstromabfragesatz "_stockQuoteSQS_" hat einen Datenstrom und eine Sicht. Datenstrom und Sicht definieren einen Namen, die Werteklasse, die SQL und den Zugriffstyp. Der Datenstrom definiert auch ein Element "basic", das angibt, dass das Attribut "volume" in der Klasse "StockQuote" der in der SQL-Anweisung definierten SQL-Spalte "transactionvolume" zugeordnet ist.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
 <objectGrids>
  <objectGrid name="og1">
   <backingMap name="stockQuote" readOnly="false" copyKey="true" streamRef="stockQuote"/>
   <backingMap name="last5MinuteAvgPrice" readOnly="false" copyKey="false"
    viewRef="last5MinuteAvgPrice"/>

   <streamQuerySet name="stockQuoteSQS">
    <stream
     name="stockQuote"
     valueClass="com.ibm.ws.objectgrid.streamquery.sample.guide.StockQuote"
     sql="create stream stockQuote
         keyed by t ( transactionvolume INTEGER, price DECIMAL (9,2), issue VARCHAR(100) );"
     access="FIELD">
     <basic name="volume" column="transactionvolume"/>
    </stream>

    <view
     name="last5MinuteAvgPrice"
     valueClass="com.ibm.ws.objectgrid.streamquery.sample.guide.AveragePrice"
     sql="CREATE VIEW last5MinuteAvgPrice AS SELECT issue, avg(price) as avgPrice
       FROM (SELECT * FROM stockQuote FETCH LATEST 5 MINUTES) group by issue;"
     access="FIELD"
    </view>
   </streamQuerySet>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Datei `objectGrid.xsd`

Verwenden Sie das ObjectGrid-XML-Deskriptorschema, um WebSphere eXtreme Scale zu konfigurieren.

Im Abschnitt „ObjectGrid-XML-Deskriptordatei" auf Seite 145 finden Sie Beschreibungen der Elemente und Attribute, die in der Datei `objectGrid.xsd` definiert sind. Informationen zur Spring-Datei `objectgrid.xsd` finden Sie im Abschnitt „Spring-XML-Deskriptordatei" auf Seite 279.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:cc="http://ibm.com/ws/objectgrid/config"
xmlns:dgc="http://ibm.com/ws/objectgrid/config"
elementFormDefault="qualified"
targetNamespace="http://ibm.com/ws/objectgrid/config">

 <xsd:element name="objectGridConfig">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element maxOccurs="1" minOccurs="1" name="objectGrids"
       type="dgc:objectGrids">
     <xsd:unique name="objectGridNameUnique">
      <xsd:selector xpath="dgc:objectGrid"/>
      <xsd:field xpath="@name"/>
     </xsd:unique>
    </xsd:element>
    <xsd:element maxOccurs="1" minOccurs="0" name="backingMapPluginCollections"
       type="dgc:backingMapPluginCollections"/>
  </xsd:sequence>
 </xsd:complexType>

 <xsd:key name="backingMapPluginCollectionId">
  <xsd:selector xpath="dgc:backingMapPluginCollections/dgc:
    backingMapPluginCollection"/>
  <xsd:field xpath="@id"/>
 </xsd:key>

 <xsd:keyref name="pluginCollectionRef" refer="dgc:backingMapPluginCollectionId">
  <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:backingMap"/>
  <xsd:field xpath="@pluginCollectionRef"/>
 </xsd:keyref>

 <xsd:key name="streamName">
  <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:
    streamQuerySet/dgc:stream"/>
  <xsd:field xpath="@name"/>
 </xsd:key>

 <xsd:keyref name="streamRef" refer="dgc:streamName">
  <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:backingMap"/>
  <xsd:field xpath="@streamRef"/>
 </xsd:keyref>

 <xsd:key name="viewName">
  <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:streamQuerySet/dgc:view"/>
  <xsd:field xpath="@name"/>
 </xsd:key>

 <xsd:keyref name="viewRef" refer="dgc:viewName">
  <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:backingMap"/>
  <xsd:field xpath="@viewRef"/>
 </xsd:keyref>
</xsd:element>

<xsd:complexType name="objectGrids">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="1" name="objectGrid"
     type="dgc:objectGrid">
   <xsd:unique name="backingMapNameUnique">
    <xsd:selector xpath="dgc:backingMap"/>
    <xsd:field xpath="@name"/>
   </xsd:unique>
   <xsd:unique name="streamQuerySetNameUnique">
    <xsd:selector xpath="dgc:streamQuerySet"/>
    <xsd:field xpath="@name"/>
   </xsd:unique>
  </xsd:element>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="backingMapPluginCollections">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="backingMapPluginCollection"
```

```
    type="dgc:backingMapPluginCollection"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="objectGrid">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="bean" type="dgc:bean"/>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="backingMap"
     type="dgc:backingMap"/>
  <xsd:element maxOccurs="1" minOccurs="0" name="querySchema" type="dgc:querySchema"/>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="streamQuerySet"
     type="dgc:streamQuerySet">
   <xsd:unique name="stream">
    <xsd:selector xpath="dgc:stream"/>
    <xsd:field xpath="@name"/>
   </xsd:unique>
   <xsd:unique name="view">
    <xsd:selector xpath="dgc:view"/>
    <xsd:field xpath="@name"/>
   </xsd:unique>
  </xsd:element>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="authorizationMechanism" type="dgc:authorizationMechanism"
   use="optional"/>
 <xsd:attribute name="accessByCreatorOnlyMode" type="dgc:accessByCreatorOnlyMode"
   use="optional"/>
 <xsd:attribute name="securityEnabled" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="txTimeout" type="xsd:int" use="optional"/>
 <xsd:attribute name="permissionCheckPeriod" type="xsd:int" use="optional"/>
 <xsd:attribute name="entityMetadataXMLFile" type="xsd:string" use="optional"/>
 <xsd:attribute name="initialState" type="dgc:initialState" use="optional"/>
</xsd:complexType>


<xsd:complexType name="backingMap">
 <xsd:sequence>
  <xsd:element maxOccurs="1" minOccurs="0" name="timeBasedDBUpdate" type="dgc:
    timeBasedDBUpdate"/>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="readOnly" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="pluginCollectionRef" type="xsd:string" use="optional"/>
 <xsd:attribute name="preloadMode" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="lockStrategy" type="dgc:lockStrategy" use="optional"/>
 <xsd:attribute name="copyMode" type="dgc:copyMode" use="optional"/>
 <xsd:attribute name="valueInterfaceClassName" type="xsd:string" use="optional"/>
 <xsd:attribute name="numberOfBuckets" type="xsd:int" use="optional"/>
 <xsd:attribute name="nullValuesSupported" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="lockTimeout" type="xsd:int" use="optional"/>
 <xsd:attribute name="numberOfLockBuckets" type="xsd:int" use="optional"/>
 <xsd:attribute name="copyKey" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="timeToLive" type="xsd:int" use="optional"/>
 <xsd:attribute name="ttlEvictoryType" type="dgc:ttlEvictorType" use="optional"/>
 <xsd:attribute name="streamRef" type="xsd:string" use="optional"/>
 <xsd:attribute name="viewRef" type="xsd:string" use="optional"/>
 <xsd:attribute name="writeBehind" type="xsd:string" use="optional"/>
 <xsd:attribute name="evictionTriggers" type="xsd:string" use="optional"/>
 <xsd:attribute name="template" type="xsd:boolean" use="optional"/>
</xsd:complexType>

<xsd:complexType name="bean">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="property" type="dgc:property"/>
 </xsd:sequence>
 <xsd:attribute name="className" type="xsd:string" use="required"/>
 <xsd:attribute name="id" type="dgc:beanId" use="required"/>
</xsd:complexType>

<xsd:complexType name="backingMapPluginCollection">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="bean" type="dgc:bean"/>
 </xsd:sequence>
 <xsd:attribute name="id" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="property">
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="value" type="xsd:string" use="required"/>
 <xsd:attribute name="type" type="dgc:propertyType" use="required"/>
```

```
 <xsd:attribute name="description" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:simpleType name="propertyType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="java.lang.Boolean"/>
  <xsd:enumeration value="boolean"/>
  <xsd:enumeration value="java.lang.String"/>
  <xsd:enumeration value="java.lang.Integer"/>
  <xsd:enumeration value="int"/>
  <xsd:enumeration value="java.lang.Double"/>
  <xsd:enumeration value="double"/>
  <xsd:enumeration value="java.lang.Byte"/>
  <xsd:enumeration value="byte"/>
  <xsd:enumeration value="java.lang.Short"/>
  <xsd:enumeration value="short"/>
  <xsd:enumeration value="java.lang.Long"/>
  <xsd:enumeration value="long"/>
  <xsd:enumeration value="java.lang.Float"/>
  <xsd:enumeration value="float"/>
  <xsd:enumeration value="java.lang.Character"/>
  <xsd:enumeration value="char"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="beanId">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="TransactionCallBack"/>
  <xsd:enumeration value="ObjectGridEventListener"/>
  <xsd:enumeration value="SubjectSource"/>
  <xsd:enumeration value="MapAuthorization"/>
  <xsd:enumeration value="SubjectValidation"/>
  <xsd:enumeration value="ObjectGridAuthorization"/>

  <xsd:enumeration value="Loader"/>
  <xsd:enumeration value="ObjectTransformer"/>
  <xsd:enumeration value="OptimisticCallback"/>
  <xsd:enumeration value="Evictor"/>
  <xsd:enumeration value="MapEventListener"/>
  <xsd:enumeration value="MapIndexPlugin"/>

 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="copyMode">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="COPY_ON_READ_AND_COMMIT"/>
  <xsd:enumeration value="COPY_ON_READ"/>
  <xsd:enumeration value="COPY_ON_WRITE"/>
  <xsd:enumeration value="NO_COPY"/>
  <xsd:enumeration value="COPY_TO_BYTES"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="lockStrategy">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="OPTIMISTIC"/>
  <xsd:enumeration value="PESSIMISTIC"/>
  <xsd:enumeration value="NONE"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ttlEvictorType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="CREATION_TIME"/>
  <xsd:enumeration value="LAST_ACCESS_TIME"/>
      <xsd:enumeration value="LAST_UPDATE_TIME"/>
  <xsd:enumeration value="NONE"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="authorizationMechanism">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="AUTHORIZATION_MECHANISM_JAAS"/>
  <xsd:enumeration value="AUTHORIZATION_MECHANISM_CUSTOM"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="accessByCreatorOnlyMode">
 <xsd:restriction base="xsd:string">
```

```
  <xsd:enumeration value="disabled"/>
  <xsd:enumeration value="complement"/>
  <xsd:enumeration value="supersede"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="streamQuerySet">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="stream" type="dgc:stream">
   <xsd:unique name="streamBasicColumnUnique">
    <xsd:selector xpath="dgc:basic"/>
    <xsd:field xpath="@column"/>
   </xsd:unique>
  </xsd:element>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="view" type="dgc:view">
   <xsd:unique name="viewBasicColumnUnique">
    <xsd:selector xpath="dgc:basic"/>
    <xsd:field xpath="@column"/>
   </xsd:unique>
  </xsd:element>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="viewResultsToListenersOnly" type="xsd:boolean"
    default="false" use="optional"/>
 <xsd:attribute name="deployInPrimaryOnly" type="xsd:boolean" default="true"
    use="optional"/>
</xsd:complexType>

<xsd:complexType name="stream">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="basic" type="dgc:basic"/>
 </xsd:sequence>
 <xsd:attribute name="valueClass" type="xsd:string" use="required"/>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="sql" type="xsd:string" use="optional"/>
 <xsd:attribute name="access" type="cc:accessType" use="optional"/>
</xsd:complexType>

<xsd:complexType name="view">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="id" type="dgc:basic"/>
  <xsd:element element maxOccurs="unbounded" minOccurs="0" name="basic"
    type="dgc:basic"/>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="sql" type="xsd:string" use="optional"/>
 <xsd:attribute name="valueClass" type="xsd:string" use="optional"/>
 <xsd:attribute name="access" type="cc:accessType" use="optional"/>
</xsd:complexType>

<xsd:complexType name="basic">
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="column" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="id">
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="column" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="timeBasedDBUpdate">
 <xsd:attribute name="persistenceUnitName" type="xsd:string" use="optional"/>
 <xsd:attribute name="mode" type="cc:dbUpdateMode" use="optional"/>
 <xsd:attribute name="timestampField" type="xsd:string" use="optional"/>
 <xsd:attribute name="entityClass" type="xsd:string" use="required"/>
 <xsd:attribute name="jpaPropertyFactory" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:simpleType name="dbUpdateMode">
 <xsd:restriction base="xsd:string"/>
  <xsd:enumeration value="INVALIDATE_ONLY"/>
  <xsd:enumeration value="UPDATE_ONLY"/>
  <xsd:enumeration value="INSERT_UPDATE"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="querySchema">
 <xsd:sequence>
  <xsd:element maxOccurs="1" minOccurs="1" name="mapSchemas" type="dgc:mapSchemas">
   <xsd:unique name="mapNameUnique">
```

```
    <xsd:selector xpath="dgc:mapSchema"/>
    <xsd:field xpath="@mapName"/>
   </xsd:unique>
  </xsd:element>
  <xsd:element maxOccurs="1" minOccurs="0" name="relationships"
    type="dgc:relationships"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="mapSchemas">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="1" name="mapSchema"
    type="dgc:mapSchema"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="relationships">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="1" name="relationship"
    type="dgc:relationship"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="mapSchema">
 <xsd:attribute name="mapName" type="xsd:string" use="required"/>
 <xsd:attribute name="valueClass" type="xsd:string" use="required"/>
 <xsd:attribute name="primaryKeyField" type="xsd:string" use="optional"/>
 <xsd:attribute name="accessType" type="cc:accessType" use="optional"/>
</xsd:complexType>

<xsd:complexType name="relationship">
 <xsd:attribute name="source" type="xsd:string" use="required"/>
 <xsd:attribute name="target" type="xsd:string" use="required"/>
 <xsd:attribute name="relationField" type="xsd:string" use="required"/>
 <xsd:attribute name="invRelationField" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:simpleType name="accessType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="PROPERTY"/>
  <xsd:enumeration value="FIELD"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="initialState">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="OFFLINE"/>
  <xsd:enumeration value="PRELOAD"/>
  <xsd:enumeration value="ONLINE"/>
 </xsd:restriction>
</xsd:simpleType>

</xsd:schema>
```

Implementierungsrichtlinien konfigurieren

Verwenden Sie die XML-Deskriptordatei für Implementierungsrichtlinien und die XML-ObjectGrid-Deskriptordatei, um eine verteilte Topologie zu verwalten. Die Implementierungsrichtlinie wird als XML-Datei codiert, die dem eXtreme-Scale-Container bereitgestellt wird. Die Implementierungsrichtlinie enthält Informationen zu Maps, MapSets, Partitionen, Replikaten usw. Außerdem steuert sie das Verhalten für die Shard-Verteilung.

Verteilte Implementierungen konfigurieren

Verwenden Sie die XML-Deskriptordatei für Implementierungsrichtlinien und die XML-ObjectGrid-Deskriptordatei, um Ihre Topologie zu verwalten.

Die Implementierungsrichtlinie wird als XML-Datei codiert, die dem eXtreme-Scale-Container bereitgestellt wird. Die XML-Datei enthält die folgenden Informationen:

- die Maps, die zu den einzelnen MapSets gehören,

- die Anzahl der Partitionen,
- die Anzahl synchroner und asynchroner Replikate.

Informationen zum Starten von Containerservern finden Sie unter eXtreme-Scale-Container automatisch starten bzw. Containerprozesse starten.

Die Implementierungsrichtlinie steuert auch die folgenden Verteilungsverhalten:
- die Mindestanzahl aktiver Container, bevor die Verteilung stattfindet,
- die automatische Verteilung verloren gegangener Shards,
- die Verteilung jedes Shards einer einzelnen Partition an eine jeweils andere Maschine.

Weitere Informationen zur Richtlinienkonfiguration finden Sie in der Beschreibung der XML-Deskriptordatei für Implementierungsrichtlinien.

Endpunktinformationen werden in der dynamischen Umgebung nicht vorkonfiguriert. Es sind keine Servernamen oder Informationen zur physischen Topologie in der Implementierungsrichtlinie enthalten. Alle Shards in einem Grid werden automatisch vom Katalogservice an die Container verteilt. Der Katalogservice verwendet die in der Implementierungsrichtlinie definierten Vorgaben, um die Verteilung der Shards automatisch zu verwalten. Mit dieser automatischen Shard-Verteilung lassen sich große Grids ohne großen Aufwand konfigurieren. Sie können der Umgebung bei Bedarf auch Server hinzufügen.

Einschränkung: In einer Umgebung mit WebSphere Application Server werden Stammgruppen mit mehr als 50 Membern nicht unterstützt.

Eine XML-Datei für Implementierungsrichtlinien wird während des Starts an einen eXtreme-Scale-Server übergeben. Eine Implementierungsrichtlinie muss zusammen mit einer ObjectGrid-XML-Datei verwendet werden. Die Implementierungsrichtlinie ist zum Starten eines Containers zwar nicht erforderlich, aber empfehlenswert. Die Implementierungsrichtlinie muss mit der verwendeten ObjectGrid-XML-Datei kompatibel sein. Für jedes Element "objectgridDeployment" in der Implementierungsrichtlinie müssen Sie ein entsprechendes Element "objectGrid" in die ObjectGrid-XML-Datei einfügen. Die Maps im objectgridDeployment-Element müssen mit den backingMap-Elementen in der ObjectGrid-XML konsistent sein. Jedes backingMap-Element darf nur in einem einzigen mapSet-Element referenziert werden.

Im folgenden Beispiel soll die Datei `companyGridDpReplication.xml` mit der entsprechenden Datei `companyGrid.xml` gepaart werden:

companyGridDpReplication.xml
```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org./2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

 <objectgridDeployment objectgridName="CompanyGrid">
  <mapSet name="mapSet1" numberOfPartitions="11"
   minSyncReplicas="1" maxSyncReplicas="1"
   maxAsyncReplicas="0" numInitialContainers="4">
   <map ref="Customer" />
   <map ref="Item" />
   <map ref="OrderLine" />
   <map ref="Order" />
  </mapSet>
 </objectgridDeployment>

</deploymentPolicy>
```

companyGrid.xml
```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
```

```
 <objectGrids>
<objectGrid name="CompanyGrid">
   <backingMap name="Customer" />
   <backingMap name="Item" />
   <backingMap name="OrderLine" />
   <backingMap name="Order" />
  </objectGrid>
 </objectGrids>

</objectGridConfig>
```

Die Datei `companyGridDpReplication.xml` enthält ein mapSet-Element , das in 11 Partitionen eingeteilt ist. Jede Partition muss genau ein synchrones Replikat haben. Die Anzahl der synchronen Replikate wird über die Attribute "minSyncReplicas" und "maxSyncReplicas" festgelegt. Da das Attribut "minSyncReplicas" auf 1 gesetzt ist, muss jede Partition im mapSet-Element mindestens ein verfügbares synchrones Replikat für die Verarbeitung von Schreiboperationen haben. Da "maxSyncReplicas" auf 1 gesetzt ist, darf jede Partition maximal ein einziges synchrones Replikat haben. Die Partitionen in diesem mapSet-Element haben keine asynchronen Replikate.

Das Attribut "numInitialContainers" weist den Katalogservice an, die Verteilung zu verzögern, bis vier Container für die Unterstützung dieser ObjectGrid-Instanz verfügbar sind. Das Attribut "numInitialContainers" wird ignoriert, sobald die angegebene Anzahl an Containern erreicht ist.

Die Datei `companyGridDpReplication.xml` ist nur ein Basisbeispiel. Eine Implementierungsrichtlinie bietet Ihnen jedoch die vollständige Kontrolle über Ihre eXtreme Scale-Umgebung. Lesen Sie die Informationen zur XML-Deskriptordatei für Implementierungsrichtlinien.

Verteilte Topologie

Verteilte kohärente Caches bieten eine höhere Leistung, Verfügbarkeit und Skalierbarkeit, die Sie konfigurieren können.

WebSphere eXtreme Scale führt eine automatische gleichmäßige Verteilung der Server durch. Sie können weitere Server hinzufügen, ohne WebSphere eXtreme Scale erneut starten zu müssen. Das Hinzufügen zusätzlicher Server ohne Neustart von eXtreme Scale ermöglicht Ihnen die Verwendung einfacher Implementierungen und Implementierungen in Terabytegröße, in denen Tausende von Servern benötigt werden.

Diese Implementierungstopologie ist flexibel. Mit dem Katalogservice können Sie Server hinzufügen und entfernen, um Ressourcen besser zu nutzen, ohne den gesamten Cache entfernen zu müssen. Sie können die Befehle "startOgServer" und "stopOgServer" verwenden, um Containerserver zu starten und zu stoppen. Beide Befehle erfordern die Angabe der Option "-catalogServiceEndPoints". Alle Clients der verteilten Topologie kommunizieren mit dem Katalogservice über Internet Interoperability Object Protocol (IIOP). Alle Clients verwenden die Schnittstelle "ObjectGrid", um mit Servern zu kommunizieren.

Mit den dynamischen Konfigurationsfunktionen von WebSphere eXtreme Scale können dem System Ressourcen ohne großen Aufwand hinzugefügt werden. Container enthalten die Daten, und der Katalogservice ermöglicht Clients die Kommunikation mit dem Container-Grid. Der Katalogservice leitet Anforderungen weiter, reserviert Speicherplatz in den Hostcontainern und verwaltet den Status und die Verfügbarkeit des Gesamtsystems. Clients stellen eine Verbindung zu einem Katalogservice her, rufen eine Beschreibung der Containerservertopologie ab und kom-

munizieren dann bei Bedarf direkt mit jedem Server. Wenn sich die Servertopologie ändert, weil neue Server hinzugefügt werden oder weil Server ausfallen, leitet der Katalogservice Clientanforderungen automatisch an den entsprechenden Server weiter, der die Daten enthält.

Ein Katalogservice existiert gewöhnlich in einem eigenen Grid von Java Virtual Machines. Ein einziger Katalogserver kann mehrere Server verwalten. Sie können einen Container in einer JVM eigenständig starten oder den Container zusammen mit anderen Containern für andere Server in eine beliebige JVM laden. Ein Client kann in jeder JVM ausgeführt werden und mit einem oder mehreren Servern kommunizieren. Ein Client kann auch in derselben JVM wie ein Container ausgeführt werden.

Sie können eine Implementierungsrichtlinie auch über das Programm erstellen, wenn Sie einen Container in einen vorhandenen Java-Prozess oder in eine vorhandene Anwendung integrieren. Weitere Informatinen finden Sie in der Dokumentation zur eXtreme-Scale-API DeploymentPolicy.

Zonen für die Verteilung von Replikaten konfigurieren

Die Zonenunterstützung ermöglicht fortgeschrittene Konfigurationen für die Verteilung von Replikaten auf Rechenzentren. Mit dieser Funktionalität können Grid mit Tausenden von Partitionen ohne großen Aufwand mit einer Handvoll optionaler Verteilungsregeln verwaltet werden. Ein Rechenzentrum kann auf verschiedene Stockwerke eines Gebäudes, verschiedene Gebäude oder selbst verschiedene Städte verteilt sein. Die Unterscheidungsmerkmale werden über Zonenregeln konfiguriert.

Flexibilität von Zonen

Sie können Shards auf Zonen verteilen. Diese Funktion gibt Ihnen mehr Kontrolle darüber, wie eXtreme Scale Shards in einem Grid verteilt. Java Virtual Machines, die einen eXtreme-Scale-Server enthalten, können mit einer Zonen-ID gekennzeichnet werden. Die Implementierungsdatei kann jetzt eine oder mehrere Zonenregeln enthalten, und diese Zonenregeln werden einem Shard-Typ zugeordnet. Am besten lässt sich dies anhand einer Reihe von Beispielen mit anschließenden Zusatzinformation erklären.

Verteilungszonen steuern, wie eXtreme Scale primäre Shards und Replikate zuordnet, um erweiterte Topologie zu konfigurieren.

Eine Java Virtual Machine kann mehrere Container, aber nur einen einzigen Server haben. Ein Container kann mehrere Shards aus einem einzigen ObjectGrid enthalten.

Diese Funktionalität ist hilfreich, um sicherzustellen, dass Replikate und primäre Shards für eine höhere Verfügbarkeit auf unterschiedliche Positionen oder Zonen verteilt werden. Normalerweise verteilt eXtreme Scale ein primäres Shard und ein Replikat-Shard nicht an Java Virtual Machines mit derselben IP-Adresse. Diese einfache Regel verhindert gewöhnlich, dass zwei eXtreme-Scale-Server auf demselben physischen Computer platziert werden. Möglicherweise benötigen Sie jedoch einen flexibleren Mechanismus. Sie verwenden beispielsweise zwei Blade-Gehäuse und möchten, das die primären Shards *einheitenübergreifend* auf beide Gehäuse verteilt werden und dass das Replikat jedes primären Shards nicht in demselben Gehäuse platziert wird wie das zugehörige primäre Shard.

Einheitenübergreifend verteilen bedeutet, dass die primären Shards in jeweils einer Zone und die zugehörigen Replikate in der jeweils anderen Zone platziert werden. Das primäre Shard 0 wird beispielsweise in Zone A und das synchrone Replikat 0 in Zone B platziert. Das primäre Shard 1 wird in Zone B und das synchrone Replikat 1 wird in Zone A platziert.

Der Gehäusename ist in diesem Fall der Zonenname. Alternativ können Sie Zonen nach den Stockwerken in einem Gebäude benennen und Zonen verwenden, um sicherzustellen, dass primäre Shards und Replikate derselben Daten auf unterschiedlichen Stockwerken gespeichert werden. Gebäude und Rechenzentren können ebenfalls verwendet werden. Es wurden Tests mit Zonen in Rechenzentren durchgeführt, um sicherzustellen, dass die Daten ordnungsgemäß zwischen den Rechenzentren repliziert werden. Wenn Sie den HTTP-Sitzungsmanager für eXtreme Scale verwenden, können Sie ebenfalls Zonen verwenden. Mit diesem Feature können Sie eine einzelne Webanwendung auf drei Rechenzentren verteilen und sicherstellen, dass HTTP-Sitzungen für Benutzer in den Rechenzentren repliziert werden, so dass die Sitzungen wiederhergestellt werden können, falls ein komplettes Rechenzentrum ausfällt.

WebSphere eXtreme Scale kennt den Bedarf, ein großes Grid über mehrere Datenzentren hinweg zu verwalten. Das Produkt kann sicherstellen, dass Sicherungen und primäre Shards für dieselbe Partition bei Bedarf auf unterschiedliche Rechenzentren verteilt werden. Es kann alle primären Shards im Rechenzentrum 1 platzieren und alle Replikate im Rechenzentrum 2, oder es kann die primären Shards und Replikate im Umlaufverfahren auf beide Datenzentren verteilen. Die Regeln sind so flexibel, dass zahlreiche Szenarien möglich sind. eXtreme Scale kann auch Tausende von Servern verwalten. Diese Funktionalität, kombiniert mit der vollständig automatischen Verteilung unter Berücksichtigung von Rechenzentren macht solch große Grids aus Verwaltungssicht kosteneffizient. Administratoren können ohne großen Aufwand und effizient festlegen, was sie möchten.

Als Administrator verwenden Sie Verteilungszonen, um zu steuern, wo primäre Shards und Replikat-Shards platziert werden. Auf diese Weise können fortgeschrittene Topologien mit hoher Leistung und hoher Verfügbarkeit konfiguriert werden. Wie bereits erwähnt, können Sie eine Zone für jede logische Gruppierung von eXtreme-Scale-Prozessen definieren. Diese Zonen können physischen Workstationstandorten wie Rechenzentren, Stockwerken eines Rechenzentrums oder Blade-Gehäusen entsprechen. Sie können Daten einheitenübergreifend auf Zonen verteilen, was Ihnen eine höhere Verfügbarkeit bietet, oder Sie können die primären Shards und Replikate auf verschiedene Zonen aufteilen, wenn ein fehlertoleranter Modus erforderlich ist.

eXtreme-Scale-Server einer Zone zuordnen, die nicht WebSphere Extended Deployment verwendet

Wenn eXtreme Scale mit Java Standard Edition oder einem Anwendungsserver verwendet wird, der nicht auf WebSphere Extended Deployment Version 6.1 basiert, kann eine JVM, die ein Shard-Container ist, mit den folgenden Verfahren einer Zone zugeordnet werden.

Anwendungen, die das Script "startOgServer" verwenden

Das Script "startOgServer" wird verwendet, um eine eXtreme-Scale-Anwendung zu starten, wenn diese nicht in einen vorhandenen Server integriert ist. Mit dem Parameter **-zone** wird die Zone angegeben, die für alle Container im Server verwendet werden soll.

Zone beim Starten eines Containers über APIs angeben

Knoten von WebSphere Extended Deployment Zonen zuordnen

Wenn Sie eXtreme Scale mit JEE-Anwendungen von WebSphere Extended Deployment verwenden, können Sie die Knotengruppen von WebSphere Extended Deployment nutzen, um Server auf bestimmte Zonen zu verteilen.

In eXtreme Scale kann eine JVM nur zu einer einzigen Zone gehören. WebSphere lässt jedoch die Zugehörigkeit eines Knotens zu mehreren Knotengruppen zu. Sie können die Funktionalität von eXtreme-Scale-Zonen verwenden, wenn Sie sicherstellen, dass jeder Ihrer Knoten nur zu einer einzigen Zonenknotengruppe gehört.

Mit der folgenden Syntax können Sie Ihre Knotengruppe benennen, um sie als Zone zu deklarieren: `Replikationszone<eindeutiges_Suffix>`. Server, die auf einem Knoten ausgeführt werden, der zu einer solchen Knotengruppe gehört, werden in die Zone eingeschlossen, die über den Namen der Knotengruppe angegeben wird. Im Folgenden finden Sie eine Beschreibung einer Beispieltopologie.

Zuerst konfigurieren Sie vier Knoten, Knoten1, Knoten2, Knoten2 und Knoten4, mit jeweils zwei Servern. Anschließend erstellen Sie eine Knotengruppe mit dem Namen "ReplikationszoneA" und eine Knotengruppe mit dem Namen "ReplikationszoneB". Danach fügen Sie Knoten1 und Knoten2 der ReplikationszoneA und Knoten3 und Knoten4 der ReplikationszoneB hinzu.

Wenn die Server auf Knoten1 und Knoten2 gestartet werden, werden sie der ReplikationszoneA zugeordnet. Knoten3 und Knoten4 werden beim Starten der ReplikationszoneB zugeordnet.

Eine Grid-Member-JVM überprüft die Zonenzugehörigkeit nur beim Start. Das Hinzufügen einer neuen Knotengruppe und das Ändern der Zugehörigkeit haben nur Auswirkungen auf neu gestartete bzw. erneut gestartete JVMs.

Zonenregeln

Eine eXtreme-Scale-Partition hat ein einziges primäres Shard und kein oder mehrere Replikat-Shards. In diesem Beispiel wird die folgende Namenskonvention für diese Shards verwendet: `P` ist das primäre Shard, `S` ist ein synchrones Replikat und `A` ein asynchrones Replikat. Eine Zonenregeln besteht aus drei Komponenten:

- Regelname,
- Zonenliste,
- Attribut inclusive oder exclusive.

Der Zonenname für einen Container kann gemäß der Beschreibung in der Dokumentation zur integrierten Server-API angegeben werden. Eine Zonenregel gibt die gültige Gruppe von Zonen an, an die ein Shard verteilt werden kann. Das Attribut "inclusive" zeigt an, dass nach der Verteilung eines Shards an eine Zone aus der Liste alle anderen Shards ebenfalls an diese Zone verteilt werden. Die Einstellung "exclusive" zeigt an, dass jedes Shard für eine Partition an eine jeweils andere Zone aus der Zonenliste verteilt wird. Die Verwendung der Einstellung "exclusive" bedeutet beispielsweise, dass die Zonenliste bei drei Shards (primäres Shard und zwei synchrone Replikate) drei Zonen enthalten muss.

Jedem Shard kann eine einzige Zonenregel zugeordnet werden. Eine Zonenregel kann von zwei Shards gemeinsam verwendet werden. Wenn eine Regel gemeinsam

genutzt wird, gilt das Attribut "inclusive" bzw. "exclusive" für die Shards aller Typen, die eine Regel gemeinsam nutzen.

Beispiele

Es folgen diverse Beispiele, die verschiedene Szenarien und die entsprechende Konfiguration für die Implementierung des jeweiligen Szenarios veranschaulichen.

Primäre Shards und Replikate einheitenübergreifend auf Zonen verteilen

Sie haben drei Blade-Gehäuse und möchten, dass die primären Shards auf alle drei Gehäuse verteilt werden und dass jeweils ein einziges Replikat auf einem jeweils anderen Gehäuse als das zugehörige primäre Shard platziert wird. Definieren Sie jedes Gehäuse als Zone mit den Gehäusenamen `ALPHA`, `BETA` und `GAMMA`. Im Folgenden sehen Sie die zugehörige Implementierungs-XML:

```
<?xml version="1.0" encoding="UTF-8"?>
 <deploymentPolicy xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="library">
   <mapSet name="ms1" numberOfPartitions="37" minSyncReplicas="1"
maxSyncReplicas="1" maxAsyncReplicas="0">
   <map ref="book" />
   <zoneMetadata>
    <shardMapping shard="P" zoneRuleRef="stripeZone"/>
    <shardMapping shard="S" zoneRuleRef="stripeZone"/>
    <zoneRule name ="stripeZone" exclusivePlacement="true" >
     <zone name="ALPHA" />
     <zone name="BETA" />
     <zone name="GAMMA" />
    </zoneRule>
   </zoneMetadata>
  </mapSet>
 </objectgridDeployment>
</deploymentPolicy>
```

Diese Implementierungs-XML enthält ein Grid mit dem Namen "library" mit einer einzigen Map mit dem Namen "book". Es werden vier Partitionen mit einem einzigen synchronen Replikat verwendet. Die Klausel für die Zonenmetadaten zeigt die Definition einer einzigen Zonenregel und die Zuordnung von Zonenregeln zu Shards. Die primären und synchronen Shards werden der Zonenregel "stripeZone" zugeordnet. Die Zonenregel umfasst alle drei Zonen und verwendet eine exklusive Verteilung. Diese Regel bedeutet Folgendes: Wenn das primäre Shard für die Partition 0 in Zone `ALPHA` platziert wird, dann wird das Replikat für die Partition 0 in Zone `BETA` oder `GAMMA` platziert. Die primären Shards für andere Partitionen werden in anderen Zonen platziert und die zugehörigen Replikate dann in einer jeweils anderen als das entsprechende primäre Shard.

Asynchrones Replikat in einer anderen Zone als das primäre Shard und das synchrone Replikat

In diesem Beispiel gibt es zwei Gebäude, die über eine Verbindung mit hoher Latenzzeit miteinander verbunden sind. In allen Szenarien möchten Sie eine hohe Verfügbarkeit, um einen Datenverlust zu vermeiden. Der Einfluss der synchronen Replikation zwischen den Gebäuden auf die Leistung drängt Sie jedoch zu einem Kompromiss. Sie möchten ein primäres Shard mit einem synchronen Replikat in einem Gebäude und ein asynchrones Replikat in einem anderen Gebäude. Normalerweise sind Ausfälle auf JVM-Abstürze oder Computerausfälle und nicht auf Probleme mit der Größe zurückzuführen. Mit dieser Topologie können Sie normale Ausfälle ohne Datenverlust bewältigen. Der Verlust eines Gebäudes tritt so selten auf, dass in diesem Fall ein gewisser Datenverlust akzeptabel ist. Sie können zwei Zonen erstellen, eine für jedes Gebäude. Im Folgenden sehen Sie die zugehörige XML-Implementierungsdatei:

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

 <objectgridDeployment objectgridName="library">
  <mapSet name="ms1" numberOfPartitions="13" minSyncReplicas="1"
   maxSyncReplicas="1" maxAsyncReplicas="1">
   <map ref="book" />
   <zoneMetadata>
    <shardMapping shard="P" zoneRuleRef="primarySync"/>
    <shardMapping shard="S" zoneRuleRef="primarySync"/>
    <shardMapping shard="A" zoneRuleRef="aysnc"/>
    <zoneRule name ="primarySync" exclusivePlacement="false" >
      <zone name="BldA" />
      <zone name="BldB" />
    </zoneRule>
    <zoneRule name="aysnc" exclusivePlacement="true">
      <zone name="BldA" />
      <zone name="BldB" />
    </zoneRule>
   </zoneMetadata>
  </mapSet>
 </objectgridDeployment>
</deploymentPolicy>
```

Das primäre Shard und das synchrone Replikat-Shard verwenden beide die Zonenregel "primaySync" mit der Einstellung "false" für das Attribut "exclusive". Wenn das primäre Shard oder das synchrone Replikat in einer Zone platziert wird, wird deshalb das jeweils andere Shard in derselben Zone platziert. Das asynchrone Replikat verwendet eine zweite Zonenregel mit denselben Zonen wie die Zonenregel "primarySync", verwendet aber die Einstellung "true" für das Attribut **exclusivePlacement**. Dieses Attribut gibt an, dass ein Shard nicht zusammen mit einem anderen Shard aus derselben Partition in einer Zone platziert werden kann. Deshalb wird das asynchrone Replikat nicht in derselben Zone platziert wie das primäre Shard bzw. das synchrone Replikat.

Alle primären Shards an eine Zone und alle Replikate an eine andere Zone verteilen

In diesem Szenario befinden sich alle primären Shards in einer bestimmten Zone und alle Replikate in einer anderen Zone. Es gibt ein primäres Shard und ein einziges asynchrones Replikat. Alle Replikate befinden sich in Zone A und alle primären Shards in Zone B.

```
<?xml version="1.0" encoding="UTF-8"?>

<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

  <objectgridDeployment objectgridName="library">
   <mapSet name="ms1" numberOfPartitions="13" minSyncReplicas="0"
    maxSyncReplicas="0" maxAsyncReplicas="1">
    <map ref="book" />
    <zoneMetadata>
     <shardMapping shard="P" zoneRuleRef="primaryRule"/>
     <shardMapping shard="A" zoneRuleRef="replicaRule"/>
     <zoneRule name ="primaryRule">
      <zone name="A" />
     </zoneRule>
     <zoneRule name="replicaRule">
      <zone name="B" />
       </zoneRule>
      </zoneMetadata>
     </mapSet>
   </objectgridDeployment>
 </deploymentPolicy>
```

In diesem Beispiel sehen Sie zwei Regeln, eine für die primären Shards (P) und eine andere für die Replikate (A).

Zonen über Weitverkehrsnetze (WANs)

Sie können eine einzige Instanz von eXtreme Scale über mehrere Gebäude oder Rechenzentren hinweg mit langsameren Netzverbindungen untereinander implementieren. Langsamere Netzverbindungen führen zu einer geringeren Bandbreite und zu einer höheren Latenzzeit. Das Risiko von Netzpartitionen ist in diesem Modus aufgrund der Netzüberlastung und anderen Faktoren ebenfalls erhöht. eXtreme Scale nähert sich diesen ungeeigneten Umgebungsbedingungen auf die folgenden Arten.

Begrenzter Austausch von Überwachungssignalen zwischen Zonen

Java Virtual Machines, die in Stammgruppen zusammengefasst sind, tauschen Überwachungssignale miteinander aus. Wenn der Katalogservice die Java Virtual Machines in Gruppen organisiert, können sich diese Gruppen nicht über mehrere Zonen erstrecken. Ein führendes Member dieser Gruppe überträgt die Zugehörigkeitsdaten mit Push an den Katalogservice. Der Katalogservice prüft alle gemeldeten Fehler, bevor er Maßnahmen ergreift. Dazu versucht er, eine Verbindung zu den fehlerverdächtigen Java Virtual Machines herzustellen. Wenn der Katalogservice eine falsche Fehlererkennung feststellt, ergreift er keine Maßnahmen, da die Stammgruppenpartition in kurzer Zeit wiederhergestellt ist.

Außerdem sendet der Katalogservice in regelmäßigen Abständen Überwachungssignale an das führende Member jeder Stammgruppe, um Fälle von Stammgruppenisolation zu behandeln.

Failover-Erkennung konfigurieren

Sie können das Intervall, in dem das System nach ausgefallenen Servern sucht, mit der Einstellung für das Intervall der Überwachungssignale konfigurieren.

Informationen zu diesem Vorgang

Die Konfiguration des Failovers variiert je nach Typ der verwendeten Umgebung. Wenn Sie eine eigenständige Umgebung verwenden, können Sie das Failover über die Befehlszeile konfigurieren. Wenn Sie eine Umgebung mit WebSphere Application Server Network Deployment verwenden, müssen Sie das Failover über die Administrationskonsole von WebSphere Application Server Network Deployment konfigurieren.

Vorgehensweise

- Failover für eigenständige Umgebungen konfigurieren

  Sie können das Intervall der Überwachungssignale über die Befehlszeile mit dem Parameter **-heartbeat** in der Scriptdatei startOgServer.bat bzw. startOgServer.sh konfigurieren. Setzen Sie den Parameter auf einen der folgenden Werte:

Tabelle 9. Intervall der Überwachungssignale

Wert	Aktion	Beschreibung
0	Typisch (Standardeinstellung)	Failover werden gewöhnlich innerhalb von 30 Sekunden erkannt.
-1	Aggressiv	Failover werden gewöhnlich innerhalb von 5 Sekunden erkannt.
1	Gelockert	Failover werden gewöhnlich innerhalb von 180 Sekunden erkannt.

Ein aggressives Intervall der Überwachungssignale kann hilfreich sein, wenn die Prozesse und das Netz stabil sind. Wenn das Netz oder die Prozesse nicht optimal konfiguriert sind, können Überwachungssignale verpasst werden, was zu einer falschen Fehlererkennung führen kann.

- Failover für Umgebungen mit WebSphere Application Server konfigurieren

  Sie können WebSphere Application Server Network Deployment Version 6.0.2 und höher so konfigurieren, dass ein schnelles Failover von WebSphere eXtreme Scale unterstützt wird. Die Standard-Failover-Zeit für permanente Fehler sind 200 Sekunden. Ein permanenter Fehler ist ein physischer Computer- oder Serverabsturz, das Ziehen des Netzkabels oder ein Betriebssystemfehler. Bei Fehlern aufgrund von Prozessabstürzen oder temporären Fehlern findet das Failover gewöhnlich in weniger als einer Sekunde statt. Die Fehlererkennung für temporäre Fehler findet statt, wenn die Netz-Sockets des inaktiven Prozesses für den Server, in dem der Prozess ausgeführt wird, automatisch vom Betriebssystem geschlossen werden.

  Überwachungssignalkonfiguration für Stammgruppen

  Wenn WebSphere eXtreme Scale in einem Prozess von WebSphere Application Server ausgeführt wird, werden die Failover-Merkmale aus den Stammgruppeneinstellungen des Anwendungsservers übernommen. In den folgenden Abschnitten wird beschrieben, wie Sie die Überwachungssignaleinstellungen der Stammgruppe für verschiedene Versionen von WebSphere Application Server Network Deployment konfigurieren:

  – **Stammgruppeneinstellungen für WebSphere Application Server Network Deployment Version 6.x und 7.x aktualisieren**

    Geben Sie das Intervall der Überwachungssignale in WebSphere Application Server Version 6.0 bis Version 6.1.0.12 in Sekunden und ab Version 6.1.0.13 in Millisekunden an. Außerdem müssen Sie die Anzahl verpasster Überwachungssignale angeben. Dieser Wert gibt an, wie viele Überwachungssignale verpasst werden können, bevor eine Peer-JVM als ausgefallen betrachtet wird. Die Erkennungszeit für permanente Fehler entspricht in etwa dem Produkt aus Intervall der Überwachungssignale und Anzahl verpasster Überwachungssignale.

    Diese Eigenschaften werden mit Hilfe von angepassten Eigenschaften in der Stammgruppe über die WebSphere-Administrationskonsole angegeben. Einzelheiten zur Konfiguration finden Sie im Abschnitt Angepasste Eigenschaften der Stammgruppe. Diese Eigenschaften müssen für alle Stammgruppen angegeben werden, die von Anwendungen verwendet werden:

    - Das Intervall der Überwachungssignale wird mit der angepassten Eigenschaft IBM_CS_FD_PERIOD_SEC (in Sekunden) bzw. der angepassten Eigenschaft IBM_CS_FD_PERIOD_MILLIS (in Millisekunden) (erfordert Version 6.1.0.13 oder höher) angegeben.
    - Die Anzahl verpasster Überwachungssignale wird mit der angepassten Eigenschaft IBM_CS_FD_CONSECUTIVE_MISSED angegeben.

    Der Standardwert für die Eigenschaft IBM_CS_FD_PERIOD_SEC ist 20, und der Standardwert für die Eigenschaft IBM_CS_FD_CONSECUTIVE_MISSED ist 10. Wenn Sie die Eigenschaft IBM_CS_FD_PERIOD_MILLIS angeben, überschreibt diese jede definierte angepasste Eigenschaft IBM_CS_FD_PERIOD_SEC. Die Werte dieser Eigenschaften sind positive ganze Zahlen.

    Verwenden Sie die folgenden Einstellungen, um eine Erkennungszeit von 1500 ms für Server der WebSphere Application Server Network Deployment Version 6.x zu erzielen:

    - Setzen Sie `IBM_CS_FD_PERIOD_MILLIS = 750` (WebSphere Application Server Network Deployment Version 6.1.0.13 und höher).

- Setzen Sie `IBM_CS_FD_CONSECUTIVE_MISSED = 2`.

– **Stammgruppeneinstellungen für WebSphere Application Server Network Deployment Version 7.0 aktualisieren**

  WebSphere Application Server Network Deployment Version 7.0 stellt zwei Stammgruppeneinstellungen bereit, die angepasst werden können, um die Failover-Erkennungszeit zu erhöhen oder zu verringern:

  - **Übertragungsintervall für Überwachungssignale.** Der Standardwert sind 30.000 Millisekunden.
  - **Überwachungssignalzeitlimit.** Der Standardwert sind 180.000 Millisekunden.

  Weitere Einzelheiten zum Ändern dieser Einstellungen finden Sie im Information Center von WebSphere Application Server Network Deployment unter "Einstellungen für die Erkennung und Fehlererkennung".

  Verwenden Sie die folgenden Einstellungen, um eine Fehlererkennungszeit von 1500 ms für Server der WebSphere Application Server Network Deployment Version 7 zu erzielen:

  - Setzen Sie das Übertragungsintervall für Überwachungssignale auf 750 Millisekunden.
  - Setzen Sie das Überwachungssignalzeitlimit auf 1500 Millisekunden.

Nächste Schritte

Wenn Sie diese Einstellungen ändern, um kürzere Failover-Zeiten anzugeben, müssen verschiedene Probleme bei der Systemoptimierung beachtet werden. Java ist keine Echtzeitumgebung. Es ist möglich, dass Threads verzögert werden, wenn die JVM lange Garbage-Collection-Zeiten verzeichnet. Threads können auch verzögert werden, wenn die Maschine, auf der die JVM ausgeführt wird, unter hoher Last steht (durch die JVM selbst oder durch andere Prozesse, die auf der Maschine ausgeführt werden). Wenn Threads verzögert werden, werden Überwachungssignale möglicherweise nicht rechtzeitig gesendet. Im schlimmsten Fall werden die durch die erforderliche Failover-Zeit verzögert. Wenn Threads verzögert werden, treten falsche Fehlererkennungen auf. Das System muss optimiert und dimensioniert werden, um sicherzustellen, dass falsche Fehlererkennungen in der Produktionsumgebung nicht auftreten. Dies kann am Zuverlässigsten durch angemessene Lasttests sichergestellt werden.

Anmerkung: Die aktuelle Version von eXtreme Scale unterstützt WebSphere Real Time.

XML-Deskriptordatei für Implementierungsrichtlinie

Zum Konfigurieren einer Implementierungsrichtlinie verwenden Sie eine XML-Deskriptordatei für die Implementierungsrichtlinie.

In den folgenden Abschnitten finden Sie Definitionen der Elemente und Attribute in der XML-Deskriptordatei für Implementierungsrichtlinien. Das entsprechende XML-Schema für die Implementierungsrichtlinie finden Sie unter „Datei `deploymentPolicy.xsd`" auf Seite 182.

Elemente in der Datei "deploymentPolicy.xml"

```
(1) <deploymentPolicy>
(2)  <objectgridDeployment objectGridName="blah">
(3)   <mapSet
(4)    name="mapSetName"
(5)    numberOfPartitions="numberOfPartitions"
(6)    minSyncReplicas="minimumNumber"
(7)     maxSyncReplicas="maximumNumber"
```

```
(8)    maxAsyncReplicas="maximumNumber"
(9)    replicaReadEnable="true|false"
(10)    numInitialContainers="numberOfInitialContainersBeforePlacement"
(11)    autoReplaceLostShards="true|false"
(12)    developmentMode="true|false"
(13)    placementStrategy="FIXED_PARTITION|PER_CONTAINER">
(14)    <map ref="backingMapReference" />
(15)   </mapSet>
(16)   <zoneMetadata>
(17)    <shardMapping
(18)      shard="shardName"
(19)      zoneRuleRef="zoneRuleRefName" />
(20)    <zoneRule
(21)      name="zoneRuleName"
(22)      exclusivePlacement="true|false" >
(23)      <zone name="ALPHA" />
(24)      <zone name="BETA" />
(25)      <zone name="GAMMA" />
(26)    </zoneRule>
(27)   </zoneMetadata>
(28)  </objectgridDeployment>
</deploymentPolicy>
```

Element "deploymentPolicy" (Zeile 1)

Das Element "deploymentPolicy" ist das Ausgangselement der XML-Datei für Implementierungsrichtlinien. Dieses Element konfiguriert den Namespace der Datei und die Schemaposition. Das Schema wird in der Datei `deploymentPolicy.xsd` definiert.

- **Anzahl der Vorkommen:** 1
- **Untergeordnetes Element:** objectgridDeployment

Element "objectgridDeployment" (Zeile 2)

Das Element "objectgridDeployment" wird verwendet, um eine ObjectGrid-Instanz aus der ObjectGrid-XML-Datei zu referenzieren. Im Element "objectgridDeployment" können Sie Ihre Maps in MapSets unterteilen.

- **Anzahl der Vorkommen:** 1 oder mehr
- **Untergeordnetes Element:** mapSet

Attribute

objectgridName
Gibt den Namen der zu implementierenden ObjectGrid-Instanz an. Dieses Attribut referenziert ein objectGrid-Element, das in der ObjectGrid-XML-Datei definiert ist. (Erforderlich)

Das Attribut "objectgridName" ist beispielsweise mit `CompanyGrid` in der Datei "companyGridDpReplication.xml" definiert. Das Attribut "objectgridName" referenziert das `CompanyGrid`, das in der Datei "companyGrid.xml" definiert ist. Lesen Sie die Informationen zur „ObjectGrid-XML-Deskriptordatei” auf Seite 145, die Sie mit der Datei für Implementierungsrichtlinien für jede ObjectGrid-Instanz koppeln müssen.

Element "mapSet" (Zeile 3)

Das Element "mapSet" wird verwendet, um Maps zu gruppieren. Die Maps in einem Element "mapSet" werden ähnlich partitioniert und repliziert. Jede Map darf nur zu einem einzigen Element "mapSet" gehören.

- **Anzahl der Vorkommen:** 1 oder mehr
- **Untergeordnetes Element:** map

Attribute

name
Gibt den Namen des MapSets an. Dieses Attribut muss innerhalb des Elements "objectgridDeployment" eindeutig sein. (Erforderlich)

numberOfPartitions
Gibt die Anzahl der Partitionen für das Element "mapSet" an. Der Standardwert ist 1. Die Anzahl muss für die Anzahl der Container, in denen die Partitionen enthalten sind, angemessen sein. (Optional)

minSyncReplicas
Gibt die Mindestanzahl synchroner Replikate für jede Partition im MapSet an. Der Standardwert ist 1. Es werden erst Shards verteilt, wenn die Domäne die Mindestanzahl synchroner Replikate unterstützen kann. Für die Unterstützung des minSyncReplicas-Werts benötigen Sie einen Container mehr, als der minSyncReplicas-Wert vorgibt. Wenn die Anzahl synchroner Replikate unter den Wert von minSyncReplicas fällt, werden keine Schreiboperationen für diese Partition mehr zugelassen. (Optional)

maxSyncReplicas
Gibt die maximale Anzahl synchroner Replikate für jede Partition im MapSet an. Der Standardwert ist 0. Es werden keine weiteren synchronen Replikate für eine Partition verteilt, wenn eine Domäne diese Anzahl synchroner Replikate für diese bestimmte Partition erreicht. Das Hinzufügen von Containern, die dieses ObjectGrid unterstützen, kann zu einer höheren Anzahl synchroner Replikate führen, wenn der maxSyncReplicas-Wert noch nicht erreicht ist. (Optional)

maxAsyncReplicas
Gibt die maximale Anzahl asynchroner Replikate für jede Partition im MapSet an. Der Standardwert ist 0. Nachdem das primäre Shard und alle synchronen Replikate für eine Partition verteilt wurden, werden asynchrone Replikate verteilt, bis der maxAsyncReplicas-Wert erreicht ist. (Optional)

replicaReadEnabled
Wenn Sie dieses Attribut auf `true` setzen, werden Leseanforderungen an die primären Shards und die Replikate einer Partition verteilt. Wenn Sie das Attribut "replicaReadEnabled" auf `false` setzen, werden Leseanforderungen nur an das primäre Shard weitergeleitet. Der Standardwert ist "false". (Optional)

numInitialContainers
Gibt die Anzahl an eXtreme-Scale-Containern an, die erforderlich sind, bevor eine Erstverteilung für die Shards in diesem mapSet-Element vorgenommen wird. Der Standardwert ist 1. Mit diesem Attribut kann Prozesszeit und Netzbandbreite eingespart werden, wenn eine ObjectGrid-Instanz online gebracht wird. (Optional)

Beim Starten eines eXtreme-Scale-Containers wird ein Ereignis an den Katalogservice gesendet. Sobald die Anzahl aktiver Container dem numInitialContainers-Wert für ein mapSet-Element entspricht, verteilt der Katalogservice die Shards aus dem MapSet, sofern der minSyncReplicas-Wert ebenfalls erreicht ist. Wenn der numInitialContainers-Wert erreicht ist, kann jedes Ereignis des Typs "Containerstart" eine Neuverteilung noch nicht verteilter und bereits verteilter Shards auslösen. Wenn Sie in etwa wissen, wie viele Container für dieses mapSet-Element gestartet werden, können Sie für "numInitialContainers" einen Wert festlegen, der dieser Zahl nahe kommt, um eine Neuverteilung nach jedem Containerstart zu vermeiden. Die Verteilung findet nur statt, wenn der im mapSet-Element angegebene numInitialContainers-Wert erreicht wird.

autoReplaceLostShards
Gibt an, ob verloren gegangene Shards an andere Container verteilt werden.

Der Standardwert ist "true".Wenn ein Container gestoppt wird oder ausfällt, gehen die Shards, die im Container ausgeführt werden, verloren. Wenn ein primäres Shard verloren geht, wird eines seiner Replikat-Shards als primäres Shard für die entsprechende Partition hochgestuft. Aufgrund dieser Hochstufung geht eines der Replikate verloren. Wenn verloren gegangene Shards nicht automatisch ersetzt werden sollen, setzen Sie das Attribut "autoReplaceLostShards" auf `false`. Diese Einstellung hat keine Auswirkung auf die Hochstufungskette, sondern nur auf die Neuverteilung des letzten Shards in der Kette. (Optional)

developmentMode
Mit diesem Attribut können Sie die Verteilung eines Shards in Relation zu dessen Peer-Shards beeinflussen. Der Standardwert ist "true". Wenn Sie das Attribut "developmentMode" auf `false` setzen, werden nie zwei Shards derselben Partition an dieselbe Maschine verteilt. Denn Sie das Attribut "developmentMode" auf `true` setzen, können Shards derselben Partition an dieselbe Maschine verteilt werden. Für beide Fälle gilt jedoch, dass zwei Shards derselben Partition nie an denselben Container verteilt werden. (Optional)

placementStrategy
Es gibt zwei Verteilungsstrategien. Die Standardstrategie ist `FIXED_PARTITION`. Bei dieser Strategie entspricht die Anzahl primärer Shards, die auf verfügbare Container verteilt wird, der Summe aus Anzahl definierter Partionen und Anzahl der Replikate. Die alternativ Strategie ist `PER_CONTAINER`. Bei dieser Strategie entspricht die Anzahl primärer Shards, die auf jeden Container verteilt wird, der Anzahl der definierten Partitionen mit einer entsprechenden Anzahl an Replikaten, die auf andere Container verteilt werden. (Optional)

Element "map" (Zeile 14)

Jedes Element "map" in einem Element "mapSet" referenziert eines der backingMap-Elemente, die in der entsprechenden ObjectGrid-XML-Datei definiert sind. Jede Map in einer verteilten Instanz von eXtreme Scale kann nur zu einem einzigen Element "mapSet" gehören.

- **Anzahl der Vorkommen:** 1 oder mehr
- **Untergeordnete Elemente:** Keine

Attribute

ref
Das Attribut enthält eine Referenz auf ein backingMap-Element in der ObjectGrid-XML-Datei. Jede Map in einem mapSet-Element muss auf ein backingMap-Element aus der ObjectGrid-XML-Datei verweisen. Der Wert des Attributs "ref" muss dem Attribut "name" eines der backingMap-Elemente in der ObjectGrid-XML-Datei entsprechen, wie im folgenden Code-Snippet gezeigt. (Erforderlich)

Element "zoneMetadata" (Zeile 16)

Sie können Shards auf Zonen verteilen. Diese Funktion gibt Ihnen mehr Kontrolle darüber, wie eXtreme Scale Shards in einem Grid verteilt. Java™ Virtual Machines, die einen eXtreme-Scale-Server enthalten, können mit einer Zonenkennung gekennzeichnet werden. Die Implementierungsdatei kann eine oder mehrere Zonenregeln enthalten, und diese Zonenregeln werden einem Shard-Typ zugeordnet. Weitere Hintergrundinformationen finden Sie unter „Zonen für die Verteilung von Replikaten konfigurieren” auf Seite 170.

- **Anzahl der Vorkommen:** 0 oder 1

- **Untergeordnete Elemente:**
  - shardMapping
  - zoneRule

Attribute: Keine

Element "shardMapping" (Zeile 17)

Jedem Shard kann eine einzige Zonenregel zugeordnet werden. Eine Zonenregel kann von zwei Shards gemeinsam verwendet werden. Wenn eine Regel gemeinsam genutzt wird, gilt das Attribut "inclusive" bzw. "exclusive" für die Shards aller Typen, die eine Regel gemeinsam nutzen.

- **Anzahl der Vorkommen:** 0 oder 1
- **Untergeordnete Elemente:** Keine

Attribute

shard
Geben Sie den Namen eines Shards an, dem die Zonenregel zugeordnet werden soll. (Erforderlich)

zoneRuleRef
Geben Sie den Namen einer Zonenregel an, die dem Shard zugeordnet werden soll. (Optional)

Element "zoneRule" (Zeile 20)

Eine Zonenregel gibt die gültige Gruppe von Zonen an, an die ein Shard verteilt werden kann.

- **Anzahl der Vorkommen:** 1 oder mehr
- **Untergeordnete Elemente:** Keine

Attribute

name
Geben Sie den Namen der zuvor definierten Zonenregel als zoneRuleRef in einem Element "shardMapping" an. (Erforderlich)

exclusivePlacement
Die Einstellung "exclusive" zeigt an, dass jedes Shard für eine Partition an eine jeweils andere Zone aus der Zonenliste verteilt wird. Die Einstellung "inclusive" zeigt an, dass nach der Verteilung eines Shards an eine Zone aus der Liste alle anderen Shards ebenfalls an diese Zone verteilt werden. Die Verwendung der Einstellung "exclusive" bedeutet beispielsweise, dass die Zonenliste bei drei Shards (primäres Shard und zwei synchrone Replikate) drei Zonen enthalten muss. (Optional)

Element "zone" (Zeilen 23 bis 25)

Angenommen, Sie haben drei Blade-Gehäuse und möchten, dass die primären Shards auf alle drei Gehäuse verteilt werden und dass jeweils ein einziges Replikat auf einem jeweils anderen Gehäuse als das zugehörige primäre Shard platziert wird. In diesem Fall können Sie jedes Gehäuse als Zone definieren und Zonennamen zuweisen, die den Gehäusenamen entsprechen: ALPHA, BETA und GAMMA.

- **Anzahl der Vorkommen:** 1 oder mehr
- **Untergeordnete Elemente:** Keine

Attribute

name
Geben Sie den Namen der Zone an, auf die Sie die angegebene Zonenregel anwenden möchten. (Erforderlich)

Beispiel

Im folgenden Beispiel wird das Element "mapSet" verwendet, um eine Implementierungsrichtlinie zu konfigurieren. Der Wert wird auf `mapSet1` gesetzt und in 10 Partitionen eingeteilt. Jede dieser Partitionen muss mindestens ein verfügbares synchrones Replikat und darf nicht mehr als zwei synchrone Replikate haben. Außerdem hat jede Partition ein asynchrones Replikat, wenn dies von der Umgebung unterstützt werden kann. Alle synchronen Replikate werden vor den asynchronen Replikaten verteilt. Der Katalogservice versucht auch erst dann, die Shards für das Element "mapSet1" zu verteilen, wenn die Domäne in der Lage ist, den minSyncReplicas-Wert zu unterstützen. Für die Unterstützung des minSyncReplicas-Werts sind mindestens zwei Container erforderlich: einer für das primäre Shard und zwei für das synchrone Replikat:

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org./2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
 xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

 <objectgridDeployment objectgridName="CompanyGrid">
  <mapSet name="mapSet1" numberOfPartitions="10"
   minSyncReplicas="1" maxSyncReplicas="2" maxAsyncReplicas="1"
    numInitialContainers="10" autoReplaceLostShards="true"
   developmentMode="false" replicaReadEnabled="true">
   <map ref="Customer"/>
   <map ref="Item"/>
   <map ref="OrderLine"/>
   <map ref="Order"/>
  </mapSet>
 </objectgridDeployment>

</deploymentPolicy>
```

Obwohl nur zwei Container erforderlich sind, um die Replikationseinstellungen zu erfüllen, erfordert das Attribut "numInitialContainers" 10 verfügbare Container, bevor der Katalogservice versucht, Shards aus diesem mapSet-Element zu verteilen. Sobald die Domäne 10 Container hat, die das ObjectGrid "CompanyGrid" unterstützen können, werden alle Shards in Element "mapSet1" verteilt.

Da das Attribut "autoReplaceLostShards" auf "true" gesetzt ist, werden alle Shards in diesem mapSet-Element, die aufgrund eines Containerausfalls verloren gegangen sind, automatisch in anderen Containern ersetzt, sofern ein Container verfügbar ist, der das verloren gegangene Shard aufnehmen kann. Shards derselben Partition können für das Element "mapSet1" nicht an dieselbe Maschine verteilt werden, weil das Attribut "developmentMode" auf `false` gesetzt ist. Reine Leseanforderungen werden auf das primäre Shard und dessen Replikate jeder Partition verteilt, weil "replicaReadEnabled" den Wert `true` hat.

Die Datei "companyGridDpMapSetAttr.xml" verwendet das Attribut "ref" in der Map, um auf die backingMap-Elemente aus der Datei "companyGrid.xml" zu verweisen.

Datei `deploymentPolicy.xsd`

Verwenden Sie das XML-Schema für Implementierungsrichtlinien, um eine XML-Deskriptordatei für Implementierungsrichtlinien zu erstellen.

Im Abschnitt „XML-Deskriptordatei für Implementierungsrichtlinie” auf Seite 177 finden Sie Beschreibungen der Elemente und Attribute, die in der Datei `deploymentPolicy.xsd` definiert sind.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:dp="http://www.ibm.com/ws/objectgrid/deploymentPolicy"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNameSpace="http://ibm.com/ws/objectgrid/deploymentPolicy"
 elementFormDefault="qualified">

 <xsd:element name="deploymentPolicy">
  <xsd:complexType>
   <xsd:choice>
    <xsd:element name="objectgridDeployment"
     type="dp:objectgridDeployment" minOccurs="1"
     maxOccurs="unbounded">
     <xsd:unique name="mapSetNameUnique">
      <xsd:selector xpath="dp:mapset" />
      <xsd:field xpath="@name" />
     </xsd:unique>
    </xsd:element>
   </xsd:choice>
  </xsd:complexType>
 </xsd:element>

 <xsd:complexType name="objectgridDeployment">
  <xsd:sequence>
   <xsd:element name="mapSet" type="dp:mapSet"
    maxOccurs="unbounded" minOccurs="1">
    <xsd:unique name="mapNameUnique">
     <xsd:selector xpath="dp:map" />
     <xsd:field xpath="@ref" />
    </xsd:unique>
   </xsd:element>
  </xsd:sequence>
  <xsd:attribute name="objectgridName" type="xsd:string"
   use="required" />
 </xsd:complexType>

 <xsd:complexType name="mapSet">
  <xsd:sequence>
   <xsd:element name="map" type="dp:map" maxOccurs="unbounded"
    minOccurs="1" />
   <xsd:element name="zoneMetadata" type="dp:zoneMetadata"
    maxOccurs="1" minOccurs="0">

    <xsd:key name="zoneRuleName">
     <xsd:selector xpath="dp:zoneRule" />
     <xsd:field xpath="@name" />
    </xsd:key>

    <xsd:keyref name="zoneRuleRef"
     refer="dp:zoneRuleName">
     <xsd:selector xpath="dp:shardMapping" />
     <xsd:field xpath="@zoneRuleRef" />
    </xsd:keyref>

   </xsd:element>
  </xsd:sequence>
  <xsd:attribute name=="name" type="xsd:string" use="required" />
  <xsd:attribute name="numberOfPartitions" type="xsd:int"
   use="optional" />
  <xsd:attribute name="minSyncReplicas" type="xsd:int"
   use="optional" />
  <xsd:attribute name="maxSyncReplicas" type="xsd:int"
   use="optional" />
  <xsd:attribute name="maxAsyncReplicas" type="xsd:int"
   use="optional" />
  <xsd:attribute name="replicaReadEnabled" type="xsd:boolean"
   use="optional" />
  <xsd:attribute name="numInitialContainers" type="xsd:int"
   use="optional" />
  <xsd:attribute name="autoReplaceLostShards" type="xsd:boolean"
   use="optional" />
  <xsd:attribute name="developmentMode" type="xsd:boolean"
   use="optional" />
  <xsd:attribute name="placementStrategy"
   type="dp:placementStrategy" use="optional" />
 </xsd:complexType>

 <xsd:simpleType name="placementStrategy">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="FIXED_PARTITIONS" />
   <xsd:enumeration value="PER_CONTAINER" />
  </xsd:restriction>
 </xsd:simpleType>


 <xsd:complexType name="map">
  <xsd:attribute name="ref" use="required" />
```

```
</xsd:complexType>

<xsd:complexType name="zoneMetadata">
 <xsd:sequence>
  <xsd:element name="shardMapping" type="dp:shardMapping"
   maxOccurs="unbounded" minOccurs="1" />
  <xsd:element name="zoneRule" type="dp:zoneRule"
   maxOccurs="unbounded" minOccurs="1">

  </xsd:element>

 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="shardMapping">
 <xsd:attribute name="shard" use="required">
  <xsd:simpleType>
   <xsd:restriction base="xsd:string">
    <xsd:enumeration value="P"></xsd:enumeration>
    <xsd:enumeration value="S"></xsd:enumeration>
    <xsd:enumeration value="A"></xsd:enumeration>
   </xsd:restriction>
  </xsd:simpleType>
 </xsd:attribute>
 <xsd:attribute name="zoneRuleRef" type="xsd:string"
  use="required" />
</xsd:complexType>

<xsd:complexType name="zoneRule">
 <xsd:sequence>
  <xsd:element name="zone" type="dp:zone"
   maxOccurs="unbounded" minOccurs="1" />
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required" />
 <xsd:attribute name="exclusivePlacement" type="xsd:boolean" />
  use="optional" />
</xsd:complexType>

<xsd:complexType name="zone">
 <xsd:attribute name="name" type="xsd:string" use="required" />
<xsd:complextType>

</xsd:schema>
```

Katalog- und Containerserver konfigurieren

Verwenden Sie eine Eigenschaftendatei, um Katalogserver und Containerserver zu konfigurieren. WebSphere eXtreme Scale hat zwei Typen von Servern: Katalogserver und Containerserver. Katalogserver steuern die Verteilung von Shards und erkennen und überwachen die Containerserver. Mehrere Katalogserver bilden den Katalogservice. Containerserver sind Java™ Virtual Machines, in denen die Anwendungsdaten für das Grid gespeichert werden.

Multimaster-Replikationstopologien konfigurieren

Mit dem Feature für asynchrone Replikation mit mehreren Mastern (Multimaster-Replikation) können Sie Links verwenden, um mehrere Domänen miteinander zu verbinden. Anschließend synchronisiert eXtreme Scale die Domänen durch Replikation über die Links. Da Sie die Links zwischen Domänen definieren, können Sie nahezu jede Topologie erstellen. Definieren Sie Links in den Eigenschaftendateien der Katalogserver für jede Domäne, oder definieren Sie links zur Laufzeit mit JMX-Programmen oder mit dem Befehlszeilendienstprogramm "xsadmin". Wenn Sie Links erstellen, wird jedoch der Satz aktueller Links für eine Domäne im Katalogservice gespeichert, was Ihnen ermöglicht, Links hinzuzufügen und zu entfernen, ohne die Domäne (das Grid) erneut zu starten.

Vorbereitende Schritte

Der Abschnitt Multimaster-Grid-Replikationstopologien (AP) enthält eine Einführung in die Merkmale verschiedener Replikationstopologien mit mehreren Mastern. Die folgende Prozedur beschreibt die Mechanismen für die Konfiguration verschiedener Links zwischen Domänen, um eine beliebige Topologie zu erstellen. Im Anschluss an die Prozedur finden Sie verschiedene Beispiele, die veranschaulichen,

wie bestimmte Topologien wie Hub- und Peripherieformationen konfiguriert werden.

Vorgehensweise

1. Definieren Sie für Bootstrap-Zwecke Links in der Eigenschaftendatei für den Katalogserver jeder Domäne in der Topologie.

   Die Eigenschaftendatei wird automatisch erkannt, wenn Sie sie `objectGridServer.properties` nennen (auf einigen Systemen muss die Groß-/Kleinschreibung beachtet werden) und in dem Klassenpfad speichern, der beim Starten einer Katalogserviceinstanz verwendet wird. Mit dem Parameter *-serverProps* können Sie die Position der Eigenschaftendatei auch in der Befehlszeile mit dem Script `startOgServer.bat|sh` angeben.

   Da bei Eigenschaftsnamen die Groß-/Kleinschreibung beachtet werden muss, achten Sie beim Aktualisieren der Eigenschaftendatei auf die Großschreibung.

   Name der lokalen Domäne
   Geben Sie den Namen "dieser" Domäne an, z. B. Domäne A:

   Beispiel:

   ```
   domainName=A
   ```

   Optionale Liste mit den Namen fremder Domänen

   Geben Sie die Namen "weiterer" Domänen in der Multimaster-Replikationstopologie an, z. B. Domäne B:

   ```
   foreignDomains=B
   ```

   Optionale Liste mit Endpunkten für die fremden Domänen
   Gibt die Verbindungsinformationen für die Katalogservices der fremden Domänen an, z. B. Domäne B:

   Beispiel:

   ```
   B.endPoints=hostB1:2809, hostB2:2809
   ```

   Wenn eine fremde Domäne mehrere Katalogservices hat, geben Sie sie alle an.

2. Verwenden Sie das Befehlszeilendienstprogramm "xsadmin" oder JMX-Programmierung, um Links zur Laufzeit hinzuzufügen oder zu entfernen.

   Die Links für eine Domäne werden im Katalogservice im replizierten Speicher verwaltet. Diese Gruppe von Links kann jederzeit vom Administrator geändert werden, ohne dass ein Neustart dieser Domäne oder einer anderen Domäne erforderlich ist. das Befehlszeilendienstprogramm "xsadmin" enthält mehrere Optionen für die Bearbeitung von Links.

   Das Dienstprogramm "xsadmin" stellt eine Verbindung zu einem Katalogservice und damit zu einer einzigen Domäne her. Deshalb kann xsadmin verwendet werden, um Links zwischen der Domäne, zu dem es eine Verbindung herstellt, und jeder anderen Domäne zu erstellen und zu entfernen.

   Verwenden Sie die Befehlszeile, um einen neuen Link zu erstellen. Beispiel:

   ```
   xsadmin –ch host –p 1099 –establishLink dname fdHostA:2809,fdHostB:2809
   ```

   Der Befehl erstellt einen neuen Link zwischen der "lokalen" Domäne und der fremden Domäne "dname", deren Katalogservice unter fdHostA:2809 und fdHostB:2809 ausgeführt wird. Die lokale Domäne hat eine Katalogservice-JVM mit der JMX-Adresse "host:1099". Geben Sie alle Katalogendpunkte der fremden Domäne an, so dass eine Verbindung mit Fehlertoleranz zur Domäne möglich ist. Es wird nicht empfohlen, ein einziges Host:Port-Paar für den Katalogservice der fremden Domäne zu verwenden.

Es ist unerheblich, welche lokale Katalogservice-JVM das Dienstprogramm "xsadmin" mit –ch und –p angibt. Es funktioniert jede Katalog-JVM. Wenn sich der Katalog in einem Deployment Manager von WebSphere Application Server befindet, wird gewöhnlich Port 9809 verwendet.

Die für die fremde Domäne angegebenen Ports sind keine JMX-Ports. Es sind die gewöhnlichen Ports, die für Clients von eXtreme Scale verwendet werden.

Nach dem Absetzen des Befehls zum Hinzufügen eines neuen Links weist der Katalogservice alle Container unter seiner Verwaltung an, mit der Replikation in der fremden Domäne zu beginnen. Ein Link ist nicht auf beiden Seiten erforderlich. Es muss nur auf einer Seite ein Link erstellt werden.

Verwenden Sie die Befehlszeile, um einen Link zu entfernen. Beispiel:

```
xsadmin –ch host –p 1099 –dismissLink dname
```

Der Befehl stellt eine Verbindung zum Katalogservice für eine Domäne her und weist diesen an, die Replikation in einer bestimmten Domäne zu stoppen. Ein Link muss nur auf einer Seite entfernt werden.

Beispiel

Angenommen, Sie möchten ein Setup mit zwei Domänen, Domäne A und Domäne B, konfigurieren.

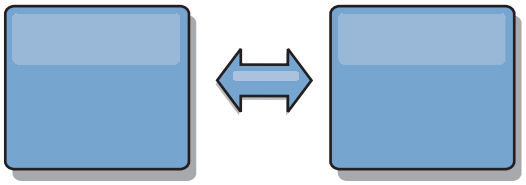


Abbildung 8. Link zwischen Domänen

Im Folgenden sehen Sie die Eigenschaftendatei für den Katalogserver in Domäne A:

```
domainName=A
foreignDomains=B
B.endPoints=hostB1:2809, hostB2:2809
```

Im Folgenden sehen Sie die Eigenschaftendatei für den Katalogserver in Domäne B. Beachten Sie die Ähnlichkeit der beiden Eigenschaftendateien.

```
domainName=B
foreignDomains=A
B.endPoints=hostB1:2809, hostB2:2809
```

Nach dem Start der beiden Domänen werden alle Grids mit den folgenden Merkmalen zwischen den Domänen repliziert.

- Hat einen privaten Katalogservice mit einem eindeutigen Domänennamen
- Hat denselben Grid-Namen wie andere Grids in der Domäne
- Hat dieselbe Anzahl an Partitionen wie andere Grids in der Domäne
- Ist ein Grid vom Typ FIXED_PARTITION (Grids vom Typ PER_CONTAINER können nicht repliziert werden)
- Hat dieselbe Anzahl an Partitionen (kann, muss aber nicht dieselbe Anzahl und dieselben Typen von Replikaten haben)
- Hat dieselben Replikationsdatentypen wie andere Grids in der Domäne

- Hat dieselben MapSet-Namen, Map-Namen und dynamischen Map-Schablonen wie andere Grids in der Domäne

Beachten Sie, dass die Replikationsrichtlinie einer Domäne ignoriert wird.

Das vorherige Beispiel zeigt, wie jede Domäne mit einem Link zur anderen Domäne konfiguriert wird, aber es muss nur in eine einzige Richtung ein Link definiert werden. Diese Tatsache ist insbesondere in Hub- und Peripherietopologien hilfreich, weil die Konfiguration sehr viel einfacher ist. Die Hub-Eigenschaftendatei muss nicht aktualisiert werden, wenn Peripheriegeräte hinzugefügt werden, und jede Peripheriegerätedatei muss nur Hub-Informationen enthalten. Auch in einer Ringtopologie muss jede Domäne nur einen Link zur vorherigen Domäne und zur nächsten Domäne im Ring haben.

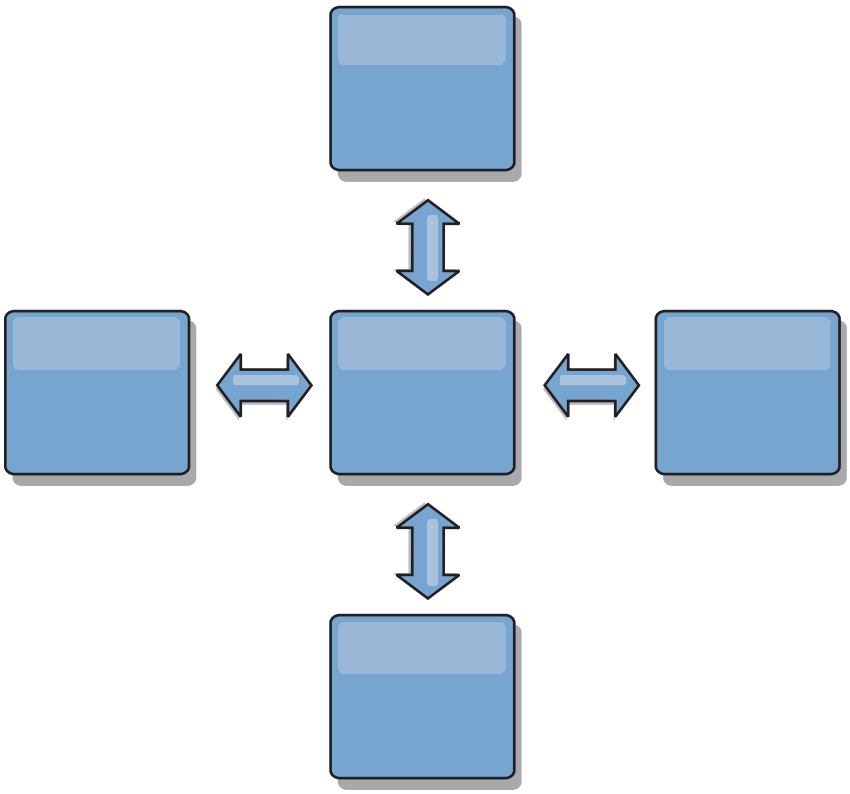


Abbildung 9. Hub- und Peripherietopologie

Der Hubs und die vier und Peripheriegeräte (Domänen A, B, C und D) haben Katalogservereigenschaftendateien wie die folgenden:

```
domainName=Hub
```

Das erste Peripheriegerät hat die folgenden Eigenschaften:

```
domainName=A
foreignDomains=Hub
Hub.endPoints=hostH1:2809, hostH2:2809
```

Das zweite Peripheriegerät hat die folgenden Eigenschaften:

```
domainName=B
foreignDomains=Hub
Hub.endPoints=hostH1:2809, hostH2:2809
```

Das dritte Peripheriegerät hat die folgenden Eigenschaften:

```
domainName=C
foreignDomains=Hub
Hub.endPoints=hostH1:2809, hostH2:2809
```

Das vierte Peripheriegerät hat die folgenden Eigenschaften:

```
domainName=D
foreignDomains=Hub
Hub.endPoints=hostH1:2809, hostH2:2809
```

Servereigenschaftendatei

Die Servereigenschaftendatei enthält verschiedene Eigenschaften, mit denen die verschiedenen Einstellungen für Ihren Server definiert werden, z. B. Trace-Einstellungen, Protokollierung und Sicherheitskonfiguration. Die Servereigenschaftendatei wird vom Katalogservice und von den Containerservern verwendet.

Musterservereigenschaftendatei

Sie können die Datei `sampleServer.properties`, die im Verzeichnis *Stammverzeichnis_von_eXtreme_Scale*`/properties` enthalten ist, verwenden, um eine eigene Eigenschaftendatei zu erstellen.

Servereigenschaftendatei angeben

Sie können die Servereigenschaftendatei mit einer der folgenden Methoden angeben. Wenn Sie eine Einstellung über einen der Einträge weiter hinten in der Liste angeben, wird die vorherige Einstellung überschrieben. Geben Sie beispielsweise einen Systemeigenschaftswert für die Servereigenschaftendatei an, überschreiben die Eigenschaften in dieser Datei die Werte in der Datei `objectGridServer.properties`, die im Klassenpfad enthalten ist.

1. Sie können irgendeine korrekt benannte Datei angeben, die im Klassenpfad enthalten ist. Wenn Sie diese korrekt benannte Datei im Arbeitsverzeichnis ablegen, wird die Datei nicht gefunden, sofern das Arbeitsverzeichnis nicht im Klassenpfad enthalten ist. Der verwendete Name lautet:

   ```
   objectGridServer.properties
   ```

2. Sie können eine Datei aus dem Systemarbeitsverzeichnis als Systemeigenschaft in einer eigenständigen Konfiguration oder in einer Konfiguration von WebSphere Application Server angeben. Die Datei darf nicht im Klassenpfad enthalten sein:

   ```
   -Dobjectgrid.server.props=Dateiname
   ```

3. Sie können die Datei als Parameter angeben, wenn Sie den Befehl "startOgServer" ausführen. Sie können diese Eigenschaften manuell überschreiben, um eine Datei im Systemarbeitsverzeichnis anzugeben.

   ```
   -serverProps Dateiname
   ```

4. Sie können die Datei als Korrekturwert mit den Methoden "ServerFactory.getServerProperties" und "ServerFactory.getCatalogServerProperties" über das Programm angeben. Das Objekt wird mit Daten aus den Eigenschaftendateien gefüllt.

Servereigenschaften

Allgemeine Eigenschaften

workingDirectory
Gibt die Position an, an die die Ausgabe des Containerservers geschrieben

wird. Wenn dieser Wert nicht angegeben wird, wird die Ausgabe in ein Verzeichnis `log` im aktuellen Verzeichnis geschrieben. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

Standardwert: Ohne

minThreads
Gibt die Mindestanzahl der Threads an, die vom internen Thread-Pool zur Laufzeit für integrierte Evictor (Bereinigungsprogramme) und DataGrid-Operationen verwendet werden.

Standardeinstellung: 10

maxThreads
Gibt die maximale Anzahl der Threads an, die vom internen Thread-Pool zur Laufzeit für integrierte Evictor (Bereinigungsprogramme) und DataGrid-Operationen verwendet werden.

Standardeinstellung: 50

traceSpec
Aktiviert die Trace-Erstellung und die Trace-Spezifikationszeichenfolge für den Containerserver. Die Trace-Erstellung ist standardmäßig inaktiviert. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

Standardwert: `*=all=disabled`

traceFile
Gibt den Namen einer Datei an, in die Trace-Informationen geschrieben werden sollen. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

systemStreamToFileEnabled
Ermöglicht dem Server, die SystemOut-, SystemErr- und Trace-Ausgabe in eine Datei zu schreiben. Wenn Sie diese Eigenschaft auf `false` setzen, wird die Ausgabe nicht in eine Datei geschrieben, sondern in der Konsole ausgegeben.

Standardwert: `true`

enableMBeans
Aktiviert Managed Beans (MBean) des ObjectGrid-Containers. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

Standardwert: `true`

serverName
Legt den Servernamen für die Identifikation des Servers fest. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

zoneName
Legt den Namen der Zone fest, zu der der Server gehört. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

haManagerPort
Synonym mit Peer-Port. Gibt die Nummer des vom High Availability Manager verwendeten Ports an. Wenn Sie diese Eigenschaft nicht definieren, generiert der Katalogservice automatisch einen verfügbaren Port. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

listenerHost
Gibt den Namen des Hosts an, zu dem der Object Request Broker (ORB) eine Bindung herstellen soll. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

Wenn Ihre Konfiguration mehrere Netzkarten enthält, definieren Sie den Listener-Host und Listener-Port, so dass der Object Request Broker in der JVM die IP-Adresse für die Bindung kennt. Geben Sie für Katalog- und Containerserver den Listener-Host und den Listener-Port in der Servereigenschaftendatei an. Wenn die zu verwendende IP-Adresse nicht angegeben wird, können Symptome wie Überschreitungen des Verbindungszeitlimits, ungewöhnliche API-Fehler und Blockierungen von Clients auftreten.

listenerPort
Gibt die Nummer des Ports an, zu dem der Object Request Broker (ORB) eine Bindung herstellen soll. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

JMXServicePort
Gibt die Nummer des Ports an, an dem der MBean-Server empfangsbereit sein soll. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

Eigenschaften des Containerservers

statsSpec
Gibt die Statistikspezifikation für den Containerserver an.

Beispiel:

```
all=disabled
```

memoryThresholdPercentage
Legt den Speicherschwellenwert für die speicherbasierte Bereinigung fest. Dieser Wert gibt an, bei welchem Prozentsatz der maximalen Heap-Speicherbelegung in der Java Virtual Machine (JVM) die Bereinigung stattfindet. Der Standardwert ist `-1`, und gibt an, dass kein Speicherschwellenwert definiert ist. Wenn Sie die Eigenschaft "memoryThresholdPercentage" definieren, wird der MemoryPoolMXBean-Wert auf der Basis des bereitgestellten Werts gesetzt. Weitere Informationen finden Sie unter MemoryPoolMXBean interface in der Java-API-Spezifikation. Die Bereinigung findet jedoch nur statt, wenn sie in einem Evictor aktiviert ist. Informationen zum Aktivieren der speicherbasierten Bereinigung finden Sie in der Beschreibung von Evictor in der *Produktübersicht*. Diese Eigenschaft gilt nur für einen Containerserver.

catalogServiceEndPoints
Gibt die Endpunkte an, die mit der Katalogservicedomäne verbunden werden sollen. Dieser Wert muss im Format `Host:Port<,Host:Port>` angegeben werden, wobei der Hostwert der listenerHost-Wert und der Portwert der listenerPort-Wert des Katalogservers sind. Diese Eigenschaft gilt nur für einen Containerserver.

Eigenschaften des Katalogservice

domainName
Gibt den Domänennamen an, der verwendet wird, um diese Katalogservicedomäne für Clients eindeutig zu identifizieren, wenn Anforderungen an mehrere Domänen verteilt werden. Diese Eigenschaft gilt nur für den Katalogservice.

enableQuorum
Aktiviert das Quorum für den Katalogservice. Das Quorum wird verwendet, um sicherzustellen, dass die Mehrheit der Member der Katalogservicedomäne verfügbar ist, bevor Änderungen an der Verteilung von Partitionen in den verfügbaren Containerservern zugelassen werden. Zum

Aktivieren des Quorums setzen Sie die Eigenschaft auf `true` oder `enabled`. Der Standardwert ist `disabled`. Diese Eigenschaft gilt nur für den Katalogservice.

catalogClusterEndpoints
Gibt die Endpunkte der Katalogservicedomäne für den Katalogservice an. Diese Eigenschaft gibt die Endpunkte des Katalogservice für das Starten der Katalogservicedomäne an. Verwenden Sie das folgende Format:

```
Servername:Hostname:Clientport:Peer-Port<Servername:Hostname:Clientport:Peer-Port>
```

Diese Eigenschaft gilt nur für den Katalogservice.

heartBeatFrequencyLevel
Gibt an, wie oft Überwachungssignale gesendet werden. Die Überwachungssignalfrequenzstufe ist ein Kompromiss zwischen Ressourcenbelegung und Fehlererkennungszeit. Je häufiger Überwachungssignale gesendet werden, desto mehr Ressourcen werden belegt, aber Fehler werden schneller erkannt. Diese Eigenschaft gilt nur für den Katalogservice.Verwenden Sie einen der folgenden Werte:

- `0`: Gibt eine typische Überwachungssignalfrequenz an. Mit diesem Wert werden Fehler in einer angemessenen Zeit ohne Ressourcenüberbelegung erkannt. (Standardwert)
- `-1`: Gibt eine aggressive Überwachungssignalfrequenz an. Mit diesem Wert werden Fehler zwar schneller erkannt, aber es werden auch mehr Prozessor- und Netzressourcen belegt. Auf dieser Stufe werden fehlende Überwachungssignale schneller erkannt, wenn der Server ausgelastet ist.
- `1`: Gibt eine gelockerte Überwachungssignalfrequenz an. Mit diesem Wert erhöht sich die Zeit für die Erkennung von Fehlern, aber es werden weniger Prozessor- und Netzressourcen belegt.

Sicherheitseigenschaften des Servers

Die Servereigenschaftendatei wird auch verwendet, um die Sicherheit des eXtreme-Scale-Servers zu konfigurieren. Sie verwenden eine einzige Servereigenschaftendatei, um die Basiseigenschaften und die Sicherheitseigenschaften zu definieren.

Allgemeine Sicherheitseigenschaften

securityEnabled
Wenn Sie diese Eigenschaft auf `true` setzen, wird die Sicherheit des Containerservers aktiviert. Der Standardwert ist `false`. Diese Eigenschaft muss der Eigenschaft "securityEnabled" entsprechen, die in der an den Katalogserver übergebenen Datei `objectGridSecurity.xml` angegeben ist.

credentialAuthentication
Gibt an, ob dieser Server die Authentifizierung von Berechtigungsnachweisen unterstützt. Wählen Sie einen der folgenden Werte aus:

- `Never`: Der Server unterstützt die Authentifizierung von Berechtigungsnachweisen nicht.
- `Supported`: Der Server unterstützt die Authentifizierung von Berechtigungsnachweisen, wenn auch der Client die Authentifizierung von Berechtigungsnachweisen unterstützt.
- `Required`: Der Client erfordert eine Authentifizierung der Berechtigungsnachweise.

Weitere Informationen zur Authentifizierung von Berechtigungsnachweisen finden Sie im Abschnitt „Anwendungsclientauthentifizierung" auf Seite 364.

Sicherheitseinstellungen auf Transportebene

transportType
Gibt den Transporttyp des Servers an. Verwenden Sie einen der folgenden Werte:

- `TCP/IP`: Gibt an, dass der Server nur TCP/IP-Verbindungen unterstützt.
- `SSL-Supported`: Gibt an, dass der Server TCP/IP- und SSL-Verbindungen (Secure Sockets Layer) unterstützt. (Standardwert)
- `SSL-Required`: Gibt an, dass der Server SSL-Verbindungen erfordert.

SSL-Konfigurationseigenschaften

Alias Gibt den Aliasnamen im Keystore an. Diese Eigenschaft wird verwendet, wenn der Keystore mehrere Schlüsselpaarzertifikate hat und Sie eines der Zertifikate auswählen möchten.

Standardeinstellung: Ohne Wert

contextProvider
Gibt den Namen des Kontextproviders für den Trust-Service an. Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme ein, die anzeigt, dass der Kontextprovidertyp ungültig ist.

Gültige Werte: `IBMJSSE2`, `IBMJSSE`, `IBMJSSEFIPS` usw.

protocol
Gibt den Typ des für den Client zu verwendenden Sicherheitsprotokolls an. Setzen Sie diesen Protokollwert auf der Basis des verwendeten JSSE-Providers (Java Secure Socket Extension). Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme ein, die anzeigt, dass der Protokollwert ungültig ist.

Gültige Werte: `SSL`, `SSLv2`, `SSLv3`, `TLS`, `TLSv1` usw.

keyStoreType
Gibt den Typ des Keystores an. Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme in der Laufzeitumgebung ein.

Gültige Werte: `JKS`, `JCEK`, `PKCS12` usw.

trustStoreType
Gibt den Typ des Truststores an. Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme in der Laufzeitumgebung ein.

Gültige Werte: `JKS`, `JCEK`, `PKCS12` usw.

keyStore
Gibt einen vollständig qualifizierten Pfad zur Keystore-Datei an.

Beispiel:

```
etc/test/security/client.private
```

trustStore
Gibt einen vollständig qualifizierten Pfad zur Truststore-Datei an.

Beispiel:

```
etc/test/security/server.public
```

keyStorePassword
Gibt die Kennwortzeichenfolge für den Keystore an. Sie können diesen Wert verschlüsseln oder den tatsächlichen Wert verwenden.

trustStorePassword
Gibt eine Kennwortzeichenfolge für den Truststore an. Sie können diesen Wert verschlüsseln oder den tatsächlichen Wert verwenden.

clientAuthentication
Wenn Sie diese Eigenschaft auf "true" setzen, muss der SSL-Client authentifiziert werden. Die Authentifizierung des SSL-Clients unterscheidet sich von der Authentifizierung des Clientzertifikats. Bei der Authentifizierung des Clientzertifikats wird ein Client auf der Basis der der Zertifizierungskette für eine Benutzer-Registry registriert. Diese Eigenschaft stellt sicher, dass der Server eine Verbindung zum richtigen Server herstellt.

Einstellung "SecureTokenManager"

Die Einstellung "SecureTokenManager" wird verwendet, um die Shared-Secret-Zeichenfolge für die gegenseitige Serverauthentifizierung und das SSO-Token (Single Sign-on) zu schützen. Weitere Informationen finden Sie im Abschnitt „Grid-Sicherheit” auf Seite 362.

secureTokenManagerType
Gibt den Typ der Einstellung "SecureTokenManager" an. Sie können eine der folgenden Einstellungen verwenden:

- `none`: Gibt an, dass kein Manager für sichere Token verwendet wird.
- `default`: Gibt an, dass der mit dem Produkt WebSphere eXtreme Scale bereitgestellte Tokenmanager verwendet wird. Sie müssen eine SecureToken-Keystore-Konfiguration angeben.
- `custom`: Gibt an, dass Sie einen eigenen Tokenmanager haben, den Sie mit der Implementierungsklasse "SecureTokenManager" angegeben haben.

customTokenManagerClass
Gibt den Namen Ihrer SecureTokenManager-Implementierungsklasse an, wenn Sie `custom` als Wert für die Eigenschaft "SecureTokenManagerType" angegeben haben. Die Implementierungsklasse muss einen zu instanziierenden Standardkonstruktor enthalten.

customSecureTokenManagerProps
Gibt die Eigenschaften der angepassten Implementierungsklasse "SecureTokenManager" an. Diese Eigenschaft wird nur verwendet, wenn die Eigenschaft "secureTokenManagerType" den Wert `custom` hat. Der Wert wird mit der Methode "setProperties(String)" im SecureTokenManager-Objekt gesetzt.

Konfiguration eines sicheren Token-Keystores

secureTokenKeyStore
Gibt den Dateipfadnamen für den Keystore an, in dem das Schlüsselpaar aus öffentlichem und privatem Schlüssel und der geheime Schlüssel gespeichert sind.

secureTokenKeyStoreType
Gibt den Keystore-Typ an, z. B. JCKES. Sie können diesen Wert auf der Basis des verwendeten JSSE-Providers (Java Secure Socket Extension) festlegen. Dieser Keystore muss jedoch geheime Schlüssel unterstützen.

secureTokenKeyPairAlias
Gibt den Alias des Schlüsselpaars aus öffentlichem und privatem Schlüssel an, das für die Signatur und Prüfung verwendet wird.

secureTokenKeyPairPassword
Gibt das Kennwort für den Schutz des Schlüsselpaars aus öffentlichem und privatem Schlüssel an, das für Signatur und Prüfung verwendet wird.

secureTokenSecretKeyAlias
Gibt den Alias des geheimen Schlüssels an, der für die Verschlüsselung verwendet wird.

secureTokenSecretKeyPassword
Gibt das Kennwort zum Schutz des geheimen Schlüssels an.

secureTokenCipherAlgorithm
Gibt den verwendeten Algorithmus für die Verschlüsselung an. Sie können diesen Wert auf der Basis des verwendeten JSSE-Providers (Java Secure Socket Extension) festlegen.

secureTokenSignAlgorithm
Gibt den für das Signieren des Objekts verwendeten Algorithmus an. Sie können diesen Wert auf der Basis des verwendeten JSSE-Providers festlegen.

Authentifizierungszeichenfolge

authenticationSecret
Gibt die Shared-Secret-Zeichenfolge für die Abfrage des Servers an. Wenn ein Server gestartet wird, muss er diese Zeichenfolge beim übergeordneten Server oder Katalogserver vorlegen. Wenn die Shared-Secret-Zeichenfolge der Zeichenfolge des übergeordneten Servers entspricht, darf der Server beitreten.

Ports konfigurieren

WebSphere eXtreme Scale ist ein verteilter Cache, der das Öffnen von Ports für die Kommunikation mit dem ORB (Object Request Broker) und dem TCP-Stack (Transmission Control Protocol) zwischen Java Virtual Machines und anderen Maschinen voraussetzt.

Netzports planen

WebSphere eXtreme Scale ist ein verteilter Cache, der das Öffnen von Ports für die Kommunikation mit dem ORB (Object Request Broker) und dem TCP-Stack (Transmission Control Protocol) zwischen Java Virtual Machines und anderen Maschinen voraussetzt. Sie sollten Ihre Ports planen und steuern, insbesondere in einer Firewall-Umgebung, z. B., wenn Sie Katalogservices und Container an mehreren Ports verwenden.

Katalogservicedomäne

Eine Katalogservicedomäne erfordert die Definition der folgenden Ports:

peerPort
Gibt den Port für den High-Availability Manager an, über den Peer-Katalogserver über einen TCP-Stack miteinander kommunizieren.

clientPort
Gibt den Port für den Zugriff auf Katalogservicedaten für Katalogserver an.

JMXServicePort
Gibt an, welchen Port der JMX-Server (Java Management Extensions) verwenden soll.

listenerPort
Definiert den ORB-Listener-Port, den Container und Clients für die Kommunikation mit dem Katalogserver über den ORB verwenden.

Wie Sie diese Ports definieren, richtet sich danach, ob Sie den eigenständigen Modus verwenden oder ob Sie die Katalogserver von eXtreme Scale in einer Umgebung von WebSphere Application Server starten:

- **Für den eigenständigen Modus:**

  Verwenden Sie den Befehl "startOgServer", um die zuvor mit der Option aufgelisteten Ports im eigenständigen Modus anzugeben, wie im folgenden Beispiel gezeigt wird:

  ```
  –catalogServiceEndPoints cs1:host1:clientPort:peerPort, cs2:host2:clientPort:peerPort,
  cs3:host3:clientPort:peerPort -listenerPort <ORB-Port> -JMXServicePort <JMX-Port>
  ```

  Weitere Informationen zum Starten der Katalogserver im eigenständigen Modus finden Sie unter „Katalogservice in einer eigenständigen Umgebung starten” auf Seite 335.

- **Für eine Umgebung mit WebSphere Application Server:**

  Sie können eine Katalogservicedomäne in der Administrationskonsole definieren. Weitere Informationen finden Sie unter „Katalogservicedomänen in WebSphere Application Server erstellen” auf Seite 350.

Containerserver

Die Containerserver von WebSphere eXtreme Scale erfordern für den Betrieb ebenfalls verschiedene Ports. Standardmäßig generiert der Containerserver von eXtreme Scale seinen HA-Manager-Port und seinen ORB-Listener-Port automatisch mit dynamischen Ports. Da es in der Firewall-Umgebung empfehlenswert ist, Ports zu planen und zu steuern, werden Optionen bereitgestellt, um Containerserver von eXtreme Scale, wie im folgenden Beispiel gezeigt, mit dem angegebenen HA-Manager-Port und dem angegebenen ORB-Listener-Port mit einer Option im Befehl "startOgServer" zu starten:

```
-HaManagerPort <Peer-Port>
-listenerPort <ORB-Port>
```

Eine ordnungsgemäße Planung der Portsteuerung ist vorteilhaft, aber die Planung und Verwaltung dieser Ports kann sich natürlich schwierig gestalten, wenn Hunderte von Java Virtual Machines in einer Maschine gestartet werden. Jeder Portkonflikt führt zu einem Fehler beim Serverstart.

Wenn die Sicherheit aktiviert ist, muss der vorherigen Portliste ein SSL-Port (Secure Sockets Layer) hinzugefügt werden. Wenn Sie `Dcom.ibm.CSI.SSLPort=<sslPort>` als Argument für **-jvmArgs** verwenden, wird der SSL-Port auf `<sslPort>` gesetzt. Als Unterstützung bei der Portplanung sollten Sie die Informationen zu den Sicherheitseinstellungen für eXtreme Scale lesen.

Ports im eigenständigen Modus konfigurieren

Eine Java Virtual Machine, in der eine Katalogserviceinstanz ausgeführt wird, erfordert vier Ports. Zwei Ports sind für interne Verwendung bestimmt, der dritte Port wird für Clients und Container-Shards für die Kommunikation mit Internet Inter-ORB Protocol (IIOP) verwendet, und der vierte Port wird für die JMX-Kommunikation (Java Management Extensions) verwendet.

Informationen zu diesem Vorgang

JVM-Endpunkte für den Katalogservice

eXtreme Scale verwendet IIOP hauptsächlich für die Kommunikation zwischen JVMs. Die JVMs des Katalogservice sind die einzigen JVMs, die eine explizite Konfiguration der Ports für die IIOP-Services und der Ports für die Gruppenservices erfordern. Die internen Ports werden mit der Befehlszeilenoption **-catalogServiceEndPoints** angegeben.

```
-catalogServiceEndPoints <Server:Host:Port:Port,Server:Host:Port:Port>
```

Mit der Befehlszeilenoption **-catalogServiceEndPoints** können Sie zwei Ports pro Server konfigurieren. Die IIOP-Ports werden mit den folgenden Befehlszeilenoptionen konfiguriert:

```
-listenerHost <Hostname>
-listenerPort <Port>
```

Geben Sie beim Starten jeder JVM des Katalogservice die vollständige Gruppe der Katalogserviceendpunkte zusammen mit einem einzigen Listener-Port für diese JVM an.

Endpunkte der Container-JVMs

Die Container-JVMs verwenden zwei Ports. Ein Port ist für die interne Verwendung bestimmt und der andere für die IIOP-Kommunikation. Die Container-JVMs suchen automatisch nicht verwendete Ports und konfigurieren sich anschließend selbst für die Verwendung der beiden dynamisch erstellten Ports. Dieser automatische Prozess minimiert die Konfiguration. Wenn jedoch Firewalls verwendet werden oder wenn Sie Ports explizit konfigurieren möchten, können Sie eine Befehlszeilenoption verwenden, um den zu verwendenden ORB-Port anzugeben:

```
-listenerHost <Hostname>
-listenerPort <Port>
```

Mit diesen Befehlszeilenoptionen können Sie den Hostnamen (wichtig für die Bindung an die richtige Netzkarte) und den Port für diese JVM angeben. Sie können den internen Port mit dem Befehlszeilenparameter **-haManagerPort** angeben. Für die einfachste Konfiguration können Sie die Ports jedoch von der Laufzeitumgebung auswählen lassen.

Vorgehensweise

1. Starten Sie den ersten Katalogserver auf hostA. Im Folgenden sehen Sie einen Beispielbefehl:

   ```
   ./startOgServer.sh cs1 -listenerHost hostA -listenerPort 2809
   -catalogServiceEndPoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
   ```

2. Starten Sie den zweiten Katalogserver auf hostB. Im Folgenden sehen Sie einen Beispielbefehl:

   ```
   ./startOgServer.sh cs2 -listenerHost hostB -listenerPort 2809
   -catalogServiceEndPoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
   ```

   Clients müssen nur die Listener-Endpunkte des Katalogservice kennen. Clients rufen Endpunkte für Container-JVMs, die die JVMs sind, die die Daten enthalten, automatisch vom Katalogservice ab. Um eine Verbindung zu dem Katalogservice aus dem vorherigen Beispiel herzustellen, muss der Client die folgende Liste von `Host:Port`-Paaren an die API "connect" übergeben:

   ```
   hostA:2809,hostB:2809
   ```

Verwenden Sie zum Starten einer Container-JVM, die den Beispielkatalogservice verwendet, den folgenden Befehl:

```
./startOgServer.sh c0 -catalogServiceEndPoints hostA:2809,hostB:2809
```

Ports in einer Umgebung mit WebSphere Application Server konfigurieren

Der Katalogservice wird standardmäßig in einer einzelnen Instanz im Deployment Manager ausgeführt und verwendet den IIOP-Bootstrap-Port (Internet Inter-ORB Protocol) für die Java Virtual Machine des Deployment Manager.

Informationen zu diesem Vorgang

Webanwendungen und EJB-Anwendungen in der Zelle können mit dem Connect-Aufruf `null,null` eine Verbindung zu Grids herstellen, die sich in derselben Zelle befinden, anstatt Bootstrap-Ports für den Katalogservice anzugeben.

Wenn die Katalogservicedomäne in WebSphere Application Server vom Deployment Manager verwaltet wird, müssen Clients außerhalb der Zelle (einschließlich Java-SE-Clients) mit dem Hostnamen des Deployment Manager und dem IIOP-Bootstrap-Port eine Verbindung zum Katalogservice herstellen. Wenn der Katalogservice in Zellen von WebSphere Application Server ausgeführt wird, während die Clients außerhalb der Zellen ausgeführt werden, suchen Sie auf den Seiten für die Konfiguration von eXtreme-Scale-Domänen in der Administrationskonsole von WebSphere Application Server nach Informationen, die erforderlich sind, um auf den Katalogservice zu verweisen. Befinden sich Ihre Clients in Zellen von WebSphere Application Server, können Sie Ports direkt von der Schnittstelle "CatalogServerProperties" abrufen.

7.1+ Obwohl Sie die Eigenschaft `catalog.services.cluster` weiterhin verwenden können, um Clientverbindungsports zu suchen, ist diese Technik veraltet. Wenn Sie diese Technik verwenden, aber den Eintrag `catalog.services.cluster` nicht finden, verwenden Sie den IIOP-Port im Deployment Manager für die Clientverbindung.

In eXtreme Scale werden die DCS-Ports (Distribution and Consistency Services) des High Availability Manager für die Gruppenzugehörigkeit wiederverwendet. Auch die JMX-Ports (Java Management Extensions) werden wiederverwendet.

Object Request Broker konfigurieren

Mit der Datei "orb.properties" können Sie die vom Object Request Broker (ORB) verwendeten Eigenschaften übergeben, um das Transportverhalten des Grids zu ändern. WebSphere® eXtreme Scale verwenden den Object Request Broker (ORB), um die Kommunikation zwischen Prozessen zu ermöglichen. Es ist keine Aktion erforderlich, um den von WebSphere eXtreme Scale oder WebSphere Application Server bereitgestellten Object Request Broker (ORB) für Ihre Server von WebSphere eXtreme Scale verwenden zu können. In diesem Abschnitt finden Sie einige Optimierungshinweise und Informationen zu weiteren ORB-bezogenen Konfigurations-Tasks, die Sie ausführen können oder müssen.

ORB-Eigenschaftendatei

Die Datei `orb.properties` wird für die Übergabe der vom Object Request Broker (ORB) verwendeten Eigenschaften verwendet, um das Transportverhalten des Grids zu ändern.

Position

Sie finden die Datei `orb.properties` im Verzeichnis `java/jre/lib`. Wenn Sie die Datei in einem Verzeichnis `java/jre/lib` von WebSphere Application Server ändern, verwenden auch die Anwendungsserver, die unter dieser Installation konfiguriert sind, die Einstellungen aus dieser Datei.

Basiseinstellungen

Die folgenden Einstellungen bilden eine gute Grundlage, sind aber nicht unbedingt die besten Einstellungen für jede Umgebung. Die Einstellungen helfen Ihnen dabei, eine gute Entscheidung bezüglich der Werte zu treffen, die in Ihrer Umgebung angemessen sind:

```
com.ibm.CORBA.RequestTimeout=30
com.ibm.CORBA.ConnectTimeout=10
com.ibm.CORBA.FragmentTimeout=30
com.ibm.CORBA.ThreadPool.MinimumSize=256
com.ibm.CORBA.ThreadPool.MaximumSize=256
com.ibm.CORBA.ThreadPool.IsGrowable=false
com.ibm.CORBA.ConnectionMultiplicity=1
com.ibm.CORBA.MinOpenConnections=1024
com.ibm.CORBA.MaxOpenConnections=1024
com.ibm.CORBA.ServerSocketQueueDepth=1024
com.ibm.CORBA.FragmentSize=0com.ibm.CORBA.iiop.NoLocalCopies=true
com.ibm.CORBA.NoLocalInterceptors=true
```

Beschreibungen der Eigenschaften

Zeitlimiteinstellungen

Die folgenden Einstellungen beziehen sich auf die Zeit, die der ORB wartet, bevor er Anforderungsoperationen aufgibt.

Anforderungszeitlimit

Eigenschaftsname: com.ibm.CORBA.RequestTimeout

Wert: Ganzzahliger Wert (in Sekunden)

Beschreibung: Gibt an, wie lange (in Sekunden) eine Anforderung auf eine Antwort warten soll, bevor sie aufgibt. Diese Eigenschaft beeinflusst die Zeit, die ein Client für das Failover benötigt, wenn ein Netzausfall eintritt. Wenn Sie einen zu niedrigen Wert für diese Eigenschaft wählen, können nicht beabsichtigte Zeitlimitüberschreitungen bei Anforderungen auftreten. Sie können dies verhindern, indem Sie den Wert für diese Eigenschaft sorgfältig auswählen.

Verbindungszeitlimit

Eigenschaftsname: com.ibm.CORBA.ConnectTimeout

Wert: Ganzzahliger Wert (in Sekunden)

Beschreibung: Gibt an, wie viele Sekunden bei einem Versuch, eine Socket-Verbindung herzustellen, gewartet werden soll, bis der Versuch eingestellt wird. Diese Eigenschaft kann wie das Anforderungszeitlimit die Zeit beeinflussen, die ein Client für das Failover benötigt, wenn ein Netzausfall eintritt. Im Allgemeinen setzen Sie diese Eigenschaft auf einen kleineren Wert als das Anforderungszeitlimit, weil die Zeit zum Herstellen einer Verbindung relativ konstant sein sollte.

Fragmentzeitlimit

Eigenschaftsname: com.ibm.CORBA.FragmentTimeout

Wert: Ganzzahliger Wert (in Sekunden)

Beschreibung: Gibt an, wie lange (in Sekunden) eine Fragmentanforderung auf eine Antwort warten soll, bevor sie aufgibt. Diese Eigenschaft ist der Eigenschaft "Anforderungszeitlimit" ähnlich.

Einstellungen des Thread-Pools

Diese Eigenschaften beschränken die Größe des Thread-Pools auf eine bestimmte Anzahl an Threads. Die Threads werden vom ORB verwendet, um die Serveranforderungen auszugliedern, nachdem sie am Socket empfangen wurden. Wenn Sie zu niedrige Werte für diese Eigenschaften wählen, kann dies zu einer erhöhten Socket-Warteschlangenlänge und zu möglichen Zeitlimitüberschreitungen führen.

Verbindungsmultiplizität

Eigenschaftsname: com.ibm.CORBA.ConnectionMultiplicity

Wert: Ganzzahliger Wert für die Anzahl der Verbindungen zwischen dem Client und dem Server. Der Standardwert ist 1. Die Festlegung eines höheren Werts definiert das Multiplexing für mehrere Verbindungen.**Beschreibung:** Ermöglicht dem ORB, mehrere Verbindungen zu jedem Server herzustellen. In der Theorie sollte die Definition dieses Werts die Parallelität der Verbindungen fördern. In der Praxis profitiert die Leistung nicht von der Festlegung der Verbindungsmultiplizität. Setzen Sie diesen Parameter nicht.

Offene Verbindungen

Eigenschaftsnamen: com.ibm.CORBA.MinOpenConnections, com.ibm-.CORBA.MaxOpenConnections

Wert: Ein ganzzahliger Wert für die Anzahl der Verbindungen.**Beschreibung:** Gibt eine Mindest- und eine maximale Anzahl offener Verbindungen an. Der ORB verwaltet einen Cache mit den Verbindungen, die mit Clients hergestellt wurden. Diese Verbindungen werden gelöscht, wenn der com.ibm.CORBA.MaxOpenConnections-Wert übergeben wird. Das Löschen von Verbindungen kann zu einem mangelhaften Verhalten im Grid führen.

Ist erweiterbar

Eigenschaftsname: com.ibm.CORBA.ThreadPool.IsGrowable

Wert: Die gültigen Werte sind `true` und `false` (Boolean).**Beschreibung:** Wenn Sie die Eigenschaft aktivieren, ist der Thread-Pool, den der ORB für eingehende Anforderungen verwendet, über die vom Pool unterstützte Größe hinaus erweiterbar. Wenn die Poolgröße überschritten wird, werden neue Threads für die Bearbeitung der Anforderungen erstellt, aber die Threads werden nicht in den Pool gestellt.

Länge der Server-Socket-Warteschlange

Eigenschaftsname: com.ibm.CORBA.ServerSocketQueueDepth

Wert: Ein ganzzahliger Wert für die Anzahl der Verbindungen.**Beschreibung:** Gibt die Länge der Warteschlange für eingehende Verbindungen von Clients an. Der ORB reiht eingehende Verbindungen von Clients in Warteschlangen ein. Wenn die Warteschlange voll ist, werden Verbindungen zurückgewiesen. Das Zurückweisen von Verbindungen kann zu einem mangelhaften Verhalten im Grid führen.

Fragmentgröße

Eigenschaftsname: com.ibm.CORBA.FragmentSize

Wert: Eine ganzzahliger Wert, der die Anzahl der Bytes angibt. Der Standardwert ist 1024.**Beschreibung:** Gibt die maximale Paketgröße an, die der ORB verwendet, wenn er eine Anforderung sendet. Wenn eine Anforderung den Grenzwert für die Fragmentgröße überschreitet, wird diese Anforderung in Anforderungsfragmente aufgeteilt, die jeweils separat gesendet und dann im Server erneut assembliert werden. Das Fragmentieren von Anforderungen ist in unzuverlässigen Netzen hilfreich, in denen Paket unter Umständen erneut gesendet werden müssen. Wenn das Netz jedoch zuverlässig ist, kann das Aufteilen von Anforderungen in Fragmente einen erhöhten Aufwand verursachen.

Keine lokalen Kopien

Eigenschaftsname: com.ibm.CORBA.iiop.NoLocalCopies

Wert: Die gültigen Werte sind `true` und `false` (Boolean). **Beschreibung:** Gibt an, ob der ORB nach Referenz übergibt. Der ORB verwendet standardmäßig Aufrufe des Typs "pass by value" (Übergeben nach Wert). Aufrufe des Typs "pass by value" verursachen zusätzliche Kosten für Garbage-Collection und Serialisierung im Pfad, wenn die Schnittstelle lokal aufgerufen wird. Wenn Sie diese Einstellung auf "true" setzen, verwendet der ORB die Methode "pass by reference" (Übergeben nach Referenz), die effizienter ist als der Aufruf des Typs "pass by value".

Keine lokalen Interceptor

Eigenschaftsname: com.ibm.CORBA.NoLocalInterceptors

Wert: Die gültigen Werte sind `true` und `false` (Boolean). **Beschreibung:** Gibt an, ob der ORB Anforderungs-Interceptor auch dann aufruft, wenn lokale Anforderungen (prozessintern) gestellt werden. Die von WebSphere eXtreme Scale verwendeten Interceptor sind für die Sicherheit und Routenverarbeitung bestimmt und nicht erforderlich, wenn die Anforderung innerhalb des Prozesses verarbeitet wird. Interceptor zwischen Prozessen sind nur für RPC-Operationen (Remote Procedure Call) erforderlich. Wenn Sie die Eigenschaft "Keine lokalen Interceptor" aktivieren, können Sie den zusätzlichen Aufwand vermeiden, den lokale Interceptor mit sich bringen.

Achtung: Wenn Sie die Sicherheit von WebSphere eXtreme Scale aktiviert haben, setzen Sie die Eigenschaft "com.ibm.CORBA.NoLocalInterceptors" auf den Wert `false`. Die Sicherheitsinfrastruktur verwendet Interceptor für die Authentifizierung.

Wenn Sie die Transportsicherheit zwischen ObjectGrid-Clients und -Servern umsetzen möchten, müssen Sie der Datei `orb.properties` weitere Eigenschaften hinzufügen. Weitere Informationen zu diesen Eigenschaften finden Sie in der Beschreibung der Verwendung der Datei `orb.properties` für die Unterstützung der Transportsicherheit im Abschnitt „Transport Layer Security und Secure Sockets Layer” auf Seite 370.

ORB-Eigenschaften und Dateideskriptoreinstellungen

Die Optimierungshinweise beziehen sich unter anderem auf die ORB-Eigenschaften (Object Request Broker) und die Dateideskriptoreinstellungen.

ORB-Eigenschaften

Der ORB (Object Request Broker) wird von WebSphere eXtreme Scale für die Kommunikation über einen TCP-Stack verwendet. Sie finden die erforderliche Datei `orb.properties` im Verzeichnis `java/jre/lib`. Für Lastspitzen mit großen Objekten aktivieren Sie die ORB-Fragmentierung mit der folgenden Einstellung:

```
com.ibm.CORBA.FragmentSize=<richtige Größe>
```

Mit der folgenden Einstellung können Sie das Anwachsen des Thread-Pools verhindern:

```
com.ibm.CORBA.ThreadPool.IsGrowable=false
```

Mit den folgenden Einstellungen können Sie angemessene Zeitlimits festlegen, um die Ausführung zu vieler Threads in einer abnormalen Situation zu verhindern:

- com.ibm.CORBA.RequestTimeout
- com.ibm.CORBA.ConnectTimeout
- com.ibm.CORBA.FragmentTimeout

Dateideskriptor

Auf UNIX®- und Linux-Systemen ist die zulässige Anzahl offener Dateien pro Prozess beschränkt. Das Betriebssystem gibt die zulässige Anzahl offener Dateien an. Wenn dieser Wert zu klein ist, tritt ein Fehler bei der Speicherzuordnung unter AIX auf, und es wird protokolliert, dass zu viele Dateien offen sind.

Setzen Sie diese Einstellung im Terminalfenster auf dem UNIX-System auf einen höheren Wert als den Systemstandardwert. Für große SMP-Maschinen mit Klonen legen Sie den Wert für eine uneingeschränkte Anzahl offener Dateien fest.

Für AIX-Konfiguration setzen Sie diese Einstellung mit dem Befehl `ulimit -n -1` auf den Wert `-1` (uneingeschränkt).

Für Solaris-Konfigurationen setzen Sie diese Einstellung mit dem Befehl `ulimit -n 16384` auf den Wert `16384`.

Zum Anzeigen des aktuellen Werts verwenden Sie den Befehl `ulimit –a`.

NIO oder ChannelFramework im ORB aktivieren

WebSphere eXtreme Scale stellt eine Servereigenschaft bereit, mit der "Transport-Mode" auf "ChannelFramework" gesetzt werden kann, um NIO (Non-blocking I/O, nicht blockierende Ein-/Ausgabe) im ORB zu aktivieren.

Vorbereitende Schritte

Suchen Sie die vorhandene Servereigenschaftendatei, oder erstellen Sie eine Servereigenschaftendatei. Sie können ChannelFramework im Katalogservice und in den Containerservern aktivieren. Weitere Einzelheiten finden Sie in der Beschreibung der Servereigenschaftendatei.

Informationen zu diesem Vorgang

Momentan können Sie mit NIO (Non-Blocking Input/Output, Nicht blockierende Ein-/Ausgabe) oder ChannelFramework in eigenständigen eXtreme-Scale-Szenarien arbeiten. Um NIO im IBM ORB zu verwenden, müssen Sie den Transportmodus des IBM ORB auf ChannelFramework setzen. Standardmäßig wird der IBM ORB

im Modus "Pluggable" ausgeführt. WebSphere eXtreme Scale stellt eine Servereigenschaft bereit, um den Transportmodus auf "ChannelFramework" zu setzen.

Wichtig:

WebSphere Application Server unterstützt den Modus "Pluggable" nur in den aktuellen Releases. Wenn WebSphere eXtreme Scale mit WebSphere Application Server integriert ausgeführt wird, muss der Modus "Pluggable" verwendet werden. Da WebSphere eXtreme Scale auch die SSL-/Transportsicherheit von WebSphere Application Server verwendet, wird deshalb momentan auch nur der Modus "Pluggable" unterstützt.

Vorgehensweise

1. Fügen Sie die Eigenschaft "enableChannelFramework=true" Ihrer Servereigenschaftendatei hinzu.
2. Stellen Sie sicher, dass die Servereigenschaftendatei der ORB-Eigenschaftendatei nicht widerspricht.

   Wenn die Servereigenschaftendatei den Transportmodus "ChannelFramework" aktiviert, aber das Attribut "TransportMode" in der Datei "orb.properties" auf "Pluggable" gesetzt ist, überschreibt der Server die Einstellung in der Datei "orb.properties" nicht. Es wird eine Warnung im Protokoll aufgezeichnet, in der darauf hingewiesen wird, dass es zwei Einstellungen gibt. Damit die Eigenschaft "enableChannelFramework=true" wirksam wird, müssen Sie die Eigenschaften anpassen, die anzeigen, dass "TransportMode" auf "Pluggable" gesetzt ist. Ändern Sie "com.ibm.CORBA.TransportMode=Pluggable" in "ChannelFramework", oder entfernen Sie die Eigenschaft.
3. Stellen Sie die aktualisierte Eigenschaftendatei beim Start des Katalogservice bzw. Containerservers bereit. Weitere Einzelheiten zur Verwendung der Servereigenschaften zum Starten eines Servers finden Sie unter Servereigenschaftendatei.

Ergebnisse

Wenn ein Katalogservice oder Containerserver den Transportmodus "channelFramework" verwendet, wird die folgende Nachricht im Protokoll ausgegeben.

```
CWOBJ0052I: Die IBM ORB-Eigenschaft TransportMode wurde auf ChannelFramework gesetzt.
```

Wenn Sie die folgende Nachricht im Protokoll sehen, überprüfen Sie, wie zuvor beschrieben, Ihre ORB-Eigenschaften.

```
CWOBJ0055W: Die IBM ORB-Eigenschaft TransportMode wurde in der Servereigenschaftendatei auf ChannelFr
gesetzt, aber die vorhandene Datei orb.properties enthält bereits eine Einstellung für TransportMode
Die TransportMode-Einstellung wird nicht überschrieben.
```

Wenn Sie ChannelFramework aktivieren, ist 512 der Maximalwert für ServerSocketQueueDepth. Wenn die Einstellung von ServerSocketQueueDepth in der Datei "orb.properties" höher als 512 ist, setzt der Server ServerSocketQueueDepth in der Datei "orb.properties" automatisch auf 512 und informiert Sie in Informationsnachricht im Protokoll darüber. Es ist keine Aktion erforderlich.

```
CWOBJ0053I: Die IBM ORB-Eigenschaft ServerSocketQueueDepth wurde auf 512
gesetzt, damit eine ordnungsgemäße Ausführung mit der TransportMode-Einstellung ChannelFramework mögl
```

Object Request Broker mit eigenständigen eXtreme-Scale-Prozessen verwenden

Sie können WebSphere eXtreme Scale mit Anwendungen verwenden, die den Object Request Broker (ORB) direkt in Umgebungen ohne WebSphere Application Server oder WebSphere Application Server Network Deployment verwenden.

Vorbereitende Schritte

Wenn Sie den ORB in demselben Prozess wie eXtreme Scale verwenden, müssen Sie bei der Ausführung von Anwendungen oder anderen Komponenten und Frameworks, die nicht mit eXtreme Scale bereitgestellt werden, möglicherweise zusätzliche Tasks ausführen, um sicherzustellen, dass eXtreme Scale ordnungsgemäß in Ihrer Umgebung ausgeführt wird.

Informationen zu diesem Vorgang

Fügen Sie der Datei `orb.properties` die Eigenschaft **ObjectGridInitializer** hinzu, um die Verwendung des ORB in Ihrer Umgebung zu initialisieren. Verwenden Sie den ORB, um die Kommunikation zwischen eXtreme-Scale-Prozessen und anderen Prozessen in Ihrer Umgebung zu aktivieren. Sie finden die Datei `orb.properties` im Verzeichnis `java/jre/lib`. Unter „ORB-Eigenschaftendatei" auf Seite 197 finden Sie Beschreibungen der Eigenschaften und Einstellungen.

Vorgehensweise

Geben Sie die folgende Zeile ein, und speichern Sie Ihre Änderungen:

```
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.objectgrid.corba.ObjectGridInitializer
```

Ergebnisse

eXtreme Scale initialisiert den ORB ordnungsgemäß und koexistiert mit anderen Anwendungen, für die der ORB aktiviert ist.

Wenn Sie eine angepasste Version des ORB mit eXtreme Scale verwenden möchten, lesen Sie den Abschnitt „Angepassten Object Request Broker konfigurieren".

Angepassten Object Request Broker konfigurieren

WebSphere eXtreme Scale verwenden den Object Request Broker (ORB), um die Kommunikation zwischen Prozessen zu ermöglichen. Es ist keine Aktion erforderlich, um den von WebSphere eXtreme Scale oder WebSphere Application Server bereitgestellten Object Request Broker (ORB) für Ihre eXtreme-Scale-Server zu verwenden. Der Aufwand für die Verwendung derselben ORBs für Ihre eXtreme-Scale-Clients zu verwenden, ist gering. Wenn Sie stattdessen einen "angepassten" ORB verwenden müssen, ist der mit IBM SDK bereitgestellte ORB eine gute Wahl, obwohl Sie gewisse Konfigurationsschritte ausführen müssen, die hier beschrieben werden. ORBs anderer Anbieter können ebenfalls mit gewissen Konfigurationsanpassungen verwendet werden.

Vorbereitende Schritte

Legen Sie fest, ob Sie den mit WebSphere eXtreme Scale oder WebSphere Application Server bereitgestellten ORB, den mit IBM SDK bereitgestellten ORB oder einen ORB eines anderen Anbieters verwenden möchten.

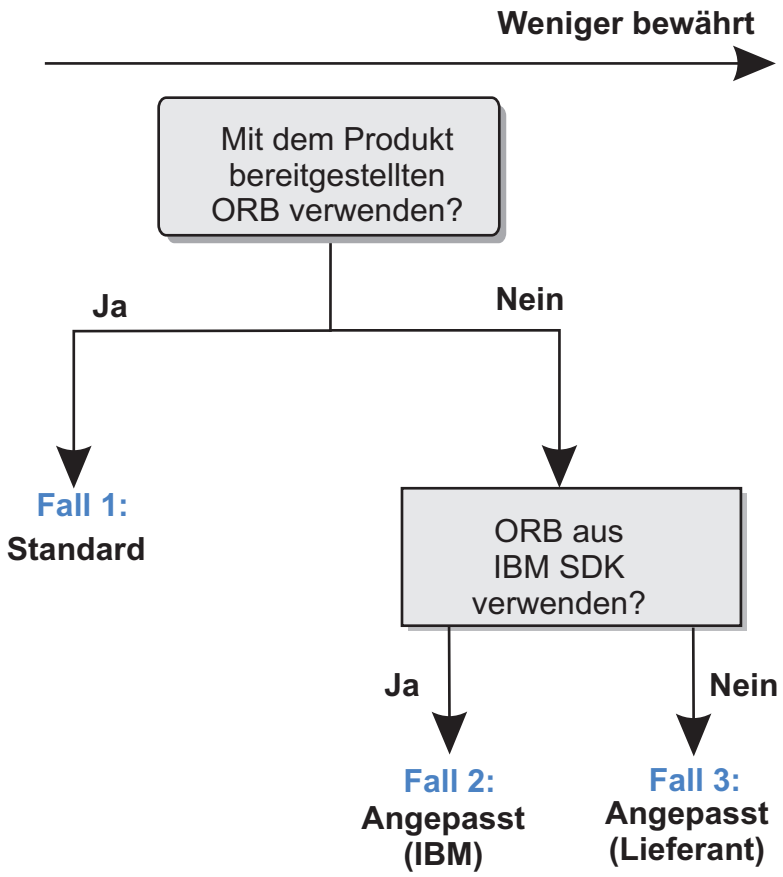


Abbildung 10. ORB auswählen

Sie können getrennte Entscheidungen für die eXtreme-Scale-Serverprozesse und die eXtreme-Scale-Clientprozesse verwenden. Obwohl eXtreme Scale Developer-Kits von den meisten Anbietern unterstützt, wird empfohlen, den ORB, der mit eXtreme Scale bereitgestellt wird, für Server- und Clientprozesse zu verwenden.eXtreme Scale bietet keine Unterstützung für den ORB, der mit dem Sun Microsystems Java Development Kit (JDK) bereitgestellt wird.

Informationen zu diesem Vorgang

Machen Sie sich mit der Konfiguration vertraut, die für die Verwendung des gewünschten ORB erforderlich ist.

Fall 1: Standard-ORB

- Für Ihre eXtreme-Scale-Serverprozesse ist zur Verwendung des mit WebSphere eXtreme Scale oder WebSphere Application Server bereitgestellten ORB keine Konfiguration erforderlich.
- Für Ihre eXtreme-Scale-Clientprozesse ist zur Verwendung des mit WebSphere eXtreme Scale oder WebSphere Application Server bereitgestellten ORB eine minimale Klassenpfadkonfiguration erforderlich.

Fall 2: Angepasster ORB (IBM)

Anweisungen zum Konfigurieren Ihrer eXtreme-Scale-Clientprozesse für die Verwendung des mit IBM SDK bereitgestellten ORB finden Sie in den Anweisungen weiter hinten in diesem Abschnitt. Sie können den IBM ORB verwenden, wenn Sie IBM SDK oder ein anderes Entwicklungskit verwenden.

Der Konfigurationsaufwand bei der Verwendung von IBM SDK Version 5 (und höher) ist geringer als bei der Verwendung von IBM SDK Version 1.4.2.

Fall 3: Angepasster ORB (andere Anbieter)
Die Verwendung eines ORB eines anderen Anbieters für Ihre eXtreme-Scale-Clientprozesse ist die Option, die am wenigsten erprobte Option. Stellen Sie sicher, dass alle Probleme, die bei der Verwendung von ORBs unabhängiger Softwareanbieter auftreten, mit dem IBM ORB und einer kompatiblen JRE reproduzierbar sind, bevor Sie Unterstützung anfordern.

Der mit Sun Microsystems Java Development Kit (JDK) bereitgestellte ORB wird nicht unterstützt.

Vorgehensweise

- Konfigurieren Sie Ihre Clientprozesse für die Verwendung eines der Standard-ORBs (**Fall 1)**.

  ```
  -jvmArgs –Djava.endorsed.dirs=Standard-ORB-Verzeichnis
  ```

- Konfigurieren Sie Client- oder Serverprozesse für die Verwendung von IBM SDK Version 5 **(Fall 2)**.
  1. Kopieren Sie die ORB-JAR-Dateien in ein leeres Verzeichnis, das nachfolgend als *angepasstes_ORB-Verzeichnis* bezeichnet wird.
     - `ibmorb.jar`
     - `ibmorbapi.jar`

     Tipp: Wenn Sie einen angepassten ORB eines anderen Anbieters verwenden (**Fall 3**), können die folgenden zusätzlichen JAR-Dateien erforderlich sein:
     - `ibmext.jar`
     - `ibmcfw.jar` bei der Verwendung des NIO-ORB
  2. Geben Sie das *angepasste_ORB-Verzeichnis* als endorsed-Verzeichnis in den Scripts an, die den Java-Befehl starten.

     Tipp: Wenn Ihre Java-Befehle bereits auf ein endorsed-Verzeichnis verweisen, muss das *angepasste_ORB-Verzeichnis* mit einer anderen Option im vorhandenen endorsed-Verzeichnis platziert werden. In diesem Fall müssen Sie die Scripts nicht aktualisieren. Wenn Sie sich trotzdem für die Aktualisierung entscheiden, müssen Sie dem vorhandenen Argument `–Djava.endorsed.dirs=` das *angepasste_ORB-Verzeichnis* voranstellen, anstatt das vorhandene Argument vollständig zu ersetzen.
     - Aktualisieren Sie Scripts für eine eigenständige eXtreme-Scale-Umgebung.

       Ändern Sie den Pfad für die Variable *OBJECTGRID_ENDORSED_DIRS* in der Datei `setupCmdLine.bat|sh` so, dass sie auf das *angepasste_ORB-Verzeichnis* zeigt. Speichern Sie Ihre Änderungen.
     - Aktualisieren Sie Scripts, wenn eXtreme Scale in eine eXtreme-Scale-Umgebung integriert ist.

       Fügen Sie die folgende Systemeigenschaft und die folgenden Parameter dem Script `startOgServer` hinzu:

       ```
       -jvmArgs –Djava.endorsed.dirs=angepasstes_ORB-Verzeichnis
       ```
     - Aktualisieren Sie angepasste Scripts, die Sie verwenden, um einen Clientanwendungsprozess oder Serverprozess zu starten.

       ```
       -Djava.endorsed.dirs=angepasstes_ORB-Verzeichnis
       ```

- Konfigurieren Sie Client- oder Serverprozesse für die Verwendung von IBM SDK Version 1.4.2 **(Fall 2)**. Wenn Ihre Umgebung ein SDK der Version 1.4.2 enthält, integrieren Sie den IBM ORB im angegebenen SDK.
  1. Laden Sie IBM SDK Version 1.4.2 herunter, und extrahieren Sie den ORB.

     Wenn kein IBM SDK für Ihre Plattform verfügbar ist, laden Sie das IBM Developer Kit für Linux, Java Technology Edition herunter, und entpacken Sie es. Weitere Informationen finden Sie auf der Webseite zu den IBM Entwicklungskits.
  2. Kopieren Sie die ORB-JAR-Dateien in das Ziel-SDK. Kopieren Sie die Dateien `java/jre/lib/ibmorb.jar` und `java/jre/lib/ibmorbapi.jar` in das Verzeichnis `java/jre/lib/ext` des Ziel-SDK.
  3. Aktualisieren Sie die ORB-Eigenschaften. Erstellen oder editieren Sie die Datei `orb.properties` im Verzeichnis `java/jre/lib` des SDK. Fügen Sie die folgenden Eigenschaften hinzu, bzw. stellen Sie sicher, dass die folgenden Eigenschaften in der Datei vorhanden sind:

     ```
     org.omg.CORBA.ORBClass=com.ibm.CORBA.iiop.ORB
     org.omg.CORBA.ORBSingletonClass=com.ibm.rmi.corba.ORBSingleton
     ```

     Beschreibungen der Eigenschaften und Einstellungen finden Sie „ORB-Eigenschaftendatei” auf Seite 197.
  4. Stellen Sie sicher, dass der XML-Parser verfügbar ist.
     - Laden Sie Xerces2 Java 2.9 von der Webseite The Apache Xerces Project - Downloads herunter.
     - Suchen Sie die Dateien `xercesImpl.jar` und `xml-apis.jar`.
     - Kopieren Sie die Dateien in das Verzeichnis `lib/ext`.

Clients konfigurieren

Sie können WebSphere® eXtreme Scale für die Ausführung in einer eigenständigen Umgebung oder für die Ausführung in einer Umgebung mit WebSphere Application Server oder WebSphere Application Server Network Deployment konfigurieren. Damit eine eXtreme-Scale-Implementierung Konfigurationsänderungen auf Server-Grid-Seite berücksichtigt, müssen Sie Prozesse erneut starten, damit diese Änderungen wirksam werden. Die Änderungen werden nicht dynamisch angewendet. Obwohl Sie auf der Clientseite die Konfigurationseinstellungen für eine vorhandene Clientinstanz nicht ändern können, können Sie jedoch unter Verwendung einer XML-Datei oder über das Programm einen neuen Client mit den erforderlichen Einstellungen erstellen. Wenn Sie einen Client erstellen, können Sie die Standardeinstellungen überschreiben, die aus der aktuellen Serverkonfiguration stammen.

Sie können einen Client von eXtreme Scale mit den folgenden Methoden konfigurieren, die jeweils mit einer XML-Datei zum Überschreiben des Clients oder über das Programm ausgeführt werden können.

- XML-Konfiguration
- Programmgesteuerte Konfiguration
- Konfiguration des Spring-Frameworks
- Nahen Cache inaktivieren

Sie können die folgenden Plug-ins in einem Client überschreiben:

- **ObjectGrid-Plug-ins**
  - TransactionCallback-Plug-in
  - ObjectGridEventListener-Plug-in

- **BackingMap-Plug-ins**
  - Evictor-Plug-in
  - MapEventListener-Plug-in
  - Attribut "numberOfBuckets"
  - Attribut "ttlEvictorType"
  - Attribut "timeToLive"

Clienteigenschaftendatei

Sie können eine Eigenschaftendatei erstellen, die auf Ihren Anforderungen für eXtreme-Scale-Clientprozesse basiert.

Musterclienteigenschaftendatei

Sie können die Datei `sampleClient.properties`, die im Verzeichnis *`Stammverzeichnis_von_eXtreme_Scale`*`\properties` enthalten ist, verwenden, um eine eigene Eigenschaftendatei zu erstellen.

Clienteigenschaftendatei angeben

Sie können die Clienteigenschaftendatei mit einer der folgenden Methoden angeben. Wenn Sie eine Einstellung über einen der Einträge weiter hinten in der Liste angeben, wird die vorherige Einstellung überschrieben. Geben Sie beispielsweise einen Systemeigenschaftswert für die Clienteigenschaftendatei an, überschreiben die Eigenschaften in dieser Datei die Werte in der Datei `objectGridClient.properties`, die im Klassenpfad enthalten ist.

1. Sie können irgendeine korrekt benannte Datei angeben, die im Klassenpfad enthalten ist. Das Ablegen dieser Datei im Systemarbeitsverzeichnis wird nicht unterstützt:

   ```
   objectGridClient.properties
   ```

2. Sie können die Datei als Systemeigenschaft in einer eigenständigen Konfiguration oder in einer Konfiguration von WebSphere Application Server angeben. Der Wert kann eine Datei im Systemarbeitsverzeichnis, aber keine Datei im Klassenpfad bezeichnen:

   ```
   -Dobjectgrid.client.props=Dateiname
   ```

3. Sie können die Datei als Korrekturwert mit der Methode "ClientClusterContext.getClientProperties" über das Programm angeben. Das Objekt wird mit Daten aus den Eigenschaftendateien gefüllt. Mit dieser Methode können keine Sicherheitseigenschaften konfiguriert werden.

Clienteigenschaften

7.1+ **listenerHost**
Gibt den Namen des Hosts an, zu dem der Object Request Broker (ORB) eine Bindung herstellen soll.

Wenn Ihre Konfiguration mehrere Netzkarten enthält, definieren Sie den Listener-Host und Listener-Port, so dass der Object Request Broker in der JVM die IP-Adresse für die Bindung kennt. Verwenden Sie für den Client die Clienteigenschaftendatei. Wenn die zu verwendende IP-Adresse nicht angegeben wird, können Symptome wie Überschreitungen des Verbindungszeitlimits, ungewöhnliche API-Fehler und Blockierungen von Clients auftreten.

7.1+ **listenerPort**
Gibt die Nummer des Ports an, zu dem der Object Request Broker (ORB) eine Bindung herstellen soll.

preferLocalProcess
Gibt an, ob der lokale Prozess für das Routing bevorzugt wird. Wenn Sie diese Eigenschaft auf "true" setzen, werden Anforderungen an Shards weitergeleitet, die sich in demselben Prozess wie der Client befinden, sofern dies angebracht ist.

Standardwert: `true`

preferLocalHost
Gibt an, ob der lokale Host für das Routing bevorzugt wird. Wenn Sie diese Eigenschaft auf "true" setzen, werden Anforderungen an Shards weitergeleitet, die sich auf demselben Host wie der Client befinden, sofern dies angebracht ist.

Standardwert: `true`

preferZones
Gibt eine Liste der bevorzugten Routing-Zonen an. Die angegebenen Zonen werden jeweils durch ein Komma voneinander getrennt:
`preferZones=ZoneA,ZoneB,ZoneC`

Standardwert: Ohne

requestRetryTimeout
Gibt an, wie lange eine Anforderung wiederholt wird (in Millisekunden). Verwenden Sie einen der folgenden gültigen Werte:

- Der Wert `0` gibt an, dass die Anforderung schnell fehlschlagen (fail-fast) und die interne Wiederholungslogik übersprungen werden soll.
- Der Wert `-1` zeigt an, dass kein Zeitlimit definiert wurde, d. h., die Anforderungsdauer wird über das Transaktionszeitlimit gesteuert. (Standardwert)
- Ein Wert größer als `0` gibt das Zeitlimit für den Anforderungseingang in Millisekunden an. Ausnahmen, bei denen auch nach einer Wiederholung der Anforderung, keine Wiederherstellung möglich ist, wie z. B. bei der Ausnahme "DuplicateException", werden sofort zurückgegeben. Das Transaktionszeitlimit wird weiterhin als maximale Wartezeit verwendet.

Sicherheitseigenschaften des Clients

Allgemeine Sicherheitseigenschaften

securityEnabled
Aktiviert die Sicherheit für eXtreme-Scale-Clients. Diese Einstellung für die Sicherheitsaktivierung muss mit der Einstellung "securityEnabled" in der Servereigenschaftendatei von WebSphere eXtreme Scale übereinstimmen. Stimmen die Einstellungen nicht überein, tritt eine Ausnahme ein.

Standardwert: `false`

Konfigurationseigenschaften für die Authentifizierung von Berechtigungsnachweisen

credentialAuthentication
Gibt die Unterstützung für die Authentifizierung von Clientberechtigungsnachweisen an. Verwenden Sie einen der folgenden gültigen Werte:

- `Never`: Der Client unterstützt die Authentifizierung von Berechtigungsnachweisen nicht.
- `Supported`: Der Client unterstützt die Authentifizierung von Berechtigungsnachweisen, wenn auch der Server die Authentifizierung von Berechtigungsnachweisen unterstützt. (Standardwert)
- `Required`: Der Client erfordert eine Authentifizierung der Berechtigungsnachweise.

authenticationRetryCount
Gibt an, wie oft die Authentifizierung wiederholt wird, wenn der Berechtigungsnachweis abgelaufen ist. Beim Wert 0 findet keine Wiederholung der Authentifizierungsversuche statt.

Standardwert: `3`

credentialGeneratorClass
Gibt den Namen der Klasse an, die die Schnittstelle "com.ibm.websphere-.objectgrid.security.plugins.CredentialGenerator" implementiert. Diese Klasse wird verwendet, um Berechtigungsnachweise für Clients abzurufen.

Standardwert: Ohne

credentialGeneratorProps
Gibt die Eigenschaften für die Implementierungsklasse "CredentialGenerator" an. Die Eigenschaften werden mit der Methode "setProperties(String)" auf das Objekt gesetzt. Der Wert "credentialGeneratorprops" wird nur verwendet, wenn der Wert der Eigenschaft "credentialGeneratorClass" ungleich null ist.

Konfigurationseigenschaften für die Sicherheit auf Transportebene

transportType
Gibt den Transporttyp des Clients an. Die gültigen Werte sind:

- `TCP/IP`: Gibt an, dass der Client nur TCP/IP-Verbindungen unterstützt.
- `SSL-Supported`: Gibt an, dass der Client TCP/IP- und SSL-Verbindungen (Secure Sockets Layer) unterstützt. (Standardwert)
- `SSL-Required`: Gibt an, dass der Client SSL-Verbindungen erfordert.

SSL-Konfigurationseigenschaften

Alias Gibt den Aliasnamen im Keystore an. Diese Eigenschaft wird verwendet, wenn der Keystore mehrere Schlüsselpaarzertifikate hat und Sie eines der Zertifikate auswählen möchten.

Standardeinstellung: Ohne Wert

contextProvider
Gibt den Namen des Kontextproviders für den Trust-Service an. Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme ein, die anzeigt, dass der Kontextprovidertyp ungültig ist.

Gültige Werte: `IBMJSSE2`, `IBMJSSE`, `IBMJSSEFIPS` usw.

protocol
Gibt den Typ des für den Client zu verwendenden Sicherheitsprotokolls an. Setzen Sie diesen Protokollwert auf der Basis des verwendeten JSSE-Providers (Java Secure Socket Extension). Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme ein, die anzeigt, dass der Protokollwert ungültig ist.

Gültige Werte: `SSL`, `SSLv2`, `SSLv3`, `TLS`, `TLSv1` usw.

keyStoreType
Gibt den Typ des Keystores an. Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme in der Laufzeitumgebung ein.

Gültige Werte: `JKS`, `JCEK`, `PKCS12` usw.

trustStoreType
Gibt den Typ des Truststores an. Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme in der Laufzeitumgebung ein.

Gültige Werte: `JKS`, `JCEK`, `PKCS12` usw.

keyStore
Gibt einen vollständig qualifizierten Pfad zur Keystore-Datei an.

Beispiel:

```
etc/test/security/client.private
```

trustStore
Gibt einen vollständig qualifizierten Pfad zur Truststore-Datei an.

Beispiel:

```
etc/test/security/server.public
```

keyStorePassword
Gibt die Kennwortzeichenfolge für den Keystore an. Sie können diesen Wert verschlüsseln oder den tatsächlichen Wert verwenden.

trustStorePassword
Gibt eine Kennwortzeichenfolge für den Truststore an. Sie können diesen Wert verschlüsseln oder den tatsächlichen Wert verwenden.

Clients mit WebSphere eXtreme Scale konfigurieren

Sie können einen eXtreme-Scale-Client Ihren Anforderungen entsprechend konfigurieren, z. B., um Einstellungen zu überschreiben.

Client mit XML konfigurieren

Zum Ändern von Einstellungen auf der Clientseite kann eine ObjectGrid-XML-Datei verwendet werden. Wenn Sie die Einstellungen in einem eXtreme-Scale-Client müssen Sie eine ObjectGrid-XML-Datei erstellen, die in ihrer Struktur der Datei gleicht, die für den eXtreme-Scale-Server verwendet wurde.

Angenommen, die folgende XML-Datei wurde mit einer XML-Deskriptordatei für Implementierungsrichtlinien kombiniert und zum Starten eines eXtreme-Scale-Servers verwendet.

companyGridServerSide.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config">

    <objectGrids>
        <objectGrid name="CompanyGrid">
            <bean id="TransactionCallback"
                className="com.company.MyTxCallback" />
            <bean id="ObjectGridEventListener"
                className="com.company.MyOgEventListener" />
            <backingMap name="Customer"
                pluginCollectionRef="customerPlugins" />
            <backingMap name="Item" />
            <backingMap name="OrderLine" numberOfBuckets="1049"
                timeToLive="1600" ttlEvictorType="LAST_ACCESS_TIME" />
```

```
            <backingMap name="Order" lockStrategy="PESSIMISTIC"
                pluginCollectionRef="orderPlugins" />
        </objectGrid>
    </objectGrids>

    <backingMapPluginCollections>
        <backingMapPluginCollection id="customerPlugins">
            <bean id="Evictor"
                className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor" />
            <bean id="MapEventListener"
                className="com.company.MyMapEventListener" />
        </backingMapPluginCollection>
        <backingMapPluginCollection id="orderPlugins">
            <bean id="MapIndexPlugin"
                className="com.company.MyMapIndexPlugin" />
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

In einem eXtreme-Scale-Server verhält sich die ObjectGrid-Instanz mit dem Namen "CompanyGrid" gemäß der Definition in der Datei "companyGridServerSide.xml". Standardmäßig hat der CompanyGrid-Client dieselben Einstellungen wie das im Server ausgeführte CompanyGrid. Einige dieser Einstellungen können jedoch wie folgt im Client überschrieben werden,

1. Erstellen Sie eine clientspezifische ObjectGrid-Instanz.
2. Kopieren Sie die ObjectGrid-XML-Datei, die zum Öffnen des Servers verwendet wurde.
3. Bearbeiten Sie die neue Datei, und passen Sie sie für die Clientseite an.
   - Zum Festlegen oder Aktualisieren von Attributen im Client geben Sie einen neuen Wert an oder ändern den vorhandenen Wert.
   - Zum Entfernen eines Plug-ins aus dem Client verwenden Sie eine leere Zeichenfolge als Wert für das Attribut "className".
   - Wenn Sie ein vorhandenes Plug-in ändern möchten, geben Sie einen neuen Wert für das Attribut "className" an.
   - Sie können auch Plug-ins hinzufügen, die für das Überschreiben eines Clients unterstützt werden: TRANSACTION_CALLBACK, OBJECTGRID_EVENT_LISTENER, EVICTOR, MAP_EVENT_LISTENER.
4. Erstellen Sie einen Client mit der neu erstellen XML-Datei für das Überschreiben des Clients:

Die folgende ObjectGrid-XML-Datei kann verwendet werden, um einige Attribute und Plug-ins im CompanyGrid-Client anzugeben:

companyGridClientSide.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
    xmlns="http://ibm.com/ws/objectgrid/config">

    <objectGrids>
        <objectGrid name="CompanyGrid">
            <bean id="TransactionCallback"
                className="com.company.MyClientTxCallback" />
            <bean id="ObjectGridEventListener" className="" />
            <backingMap name="Customer" numberOfBuckets="1429"
                pluginCollectionRef="customerPlugins" />
            <backingMap name="Item" />
            <backingMap name="OrderLine" numberOfBuckets="701"
                timeToLive="800" ttlEvictorType="LAST_ACCESS_TIME" />
            <backingMap name="Order" lockStrategy="PESSIMISTIC"
                pluginCollectionRef="orderPlugins" />
        </objectGrid>
    </objectGrids>

    <backingMapPluginCollections>
        <backingMapPluginCollection id="customerPlugins">
```

```
            <bean id="Evictor"
                className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor" />
            <bean id="MapEventListener" className="" />
        </backingMapPluginCollection>
        <backingMapPluginCollection id="orderPlugins">
            <bean id="MapIndexPlugin"
                className="com.company.MyMapIndexPlugin" />
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

- Der TransactionCallback im Client ist "com.company.MyClientTxCallback" und nicht wie auf der Serverseite "com.company.MyTxCallback".
- Der Client hat kein ObjectGridEventListener-Plug-in, weil der className-Wert die leere Zeichenfolge ist.
- Der Client setzt "numberOfBuckets" für die BackingMap "Customer" auf 1429, behält sein Evictor-Plug-in und entfernt das MapEventListener-Plug-in.
- Die Attribute "numberOfBuckets" und "timeToLive" der BackingMap "OrderLine" haben sich geändert.
- Obwohl ein anderes lockStrategy-Attribut angegeben ist, hat dieses keine Wirkung, weil das Attribut "lockStrategy" für das Überschreiben eines Clients nicht unterstützt wird.

Zum Erstellen eines CompanyGrid-Clients über die Datei "companyGridClientSide.xml" übergeben Sie die ObjectGrid-XML-Datei als URL an eine der connect-Methoden im ObjectGridManager.

Client für XML erstellen

```
ObjectGridManager ogManager =
 ObjectGridManagerFactory.ObjectGridManager();
ClientClusterContext clientClusterContext =
 ogManager.connect("MyServer1.company.com:2809", null, new URL(
                "file:xml/companyGridClientSide.xml"));
```

Client über das Programm konfigurieren

Sie können die clientseitigen ObjectGrid-Einstellungen auch über das Programm überschreiben. Erstellen Sie ein ObjectGridConfiguration-Objekt, das in der Struktur der serverseitigen ObjectGrid-Instanz gleicht. Der folgende Code erstellt eine clientseite ObjectGrid-Instanz, die funktional äquivalent zum Überschreiben des Clients im vorherigen Abschnitt ist, für as eine XML-Datei verwendet wird.

Clientseitige Einstellungen über das Programm überschreiben

```
ObjectGridConfiguration companyGridConfig = ObjectGridConfigFactory
    .createObjectGridConfiguration("CompanyGrid");
Plugin txCallbackPlugin = ObjectGridConfigFactory.createPlugin(
    PluginType.TRANSACTION_CALLBACK, "com.company.MyClientTxCallback");
companyGridConfig.addPlugin(txCallbackPlugin);

Plugin ogEventListenerPlugin = ObjectGridConfigFactory.createPlugin(
    PluginType.OBJECTGRID_EVENT_LISTENER, "");
companyGridConfig.addPlugin(ogEventListenerPlugin);

BackingMapConfiguration customerMapConfig = ObjectGridConfigFactory
    .createBackingMapConfiguration("Customer");
customerMapConfig.setNumberOfBuckets(1429);
Plugin evictorPlugin = ObjectGridConfigFactory.createPlugin(PluginType.EVICTOR,
    "com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor");
customerMapConfig.addPlugin(evictorPlugin);

companyGridConfig.addBackingMapConfiguration(customerMapConfig);

BackingMapConfiguration orderLineMapConfig = ObjectGridConfigFactory
    .createBackingMapConfiguration("OrderLine");
orderLineMapConfig.setNumberOfBuckets(701);
orderLineMapConfig.setTimeToLive(800);
orderLineMapConfig.setTtlEvictorType(TTLType.LAST_ACCESS_TIME);

companyGridConfig.addBackingMapConfiguration(orderLineMapConfig);
```

```
List ogConfigs = new ArrayList();
ogConfigs.add(companyGridConfig);

Map overrideMap = new HashMap();
overrideMap.put(CatalogServerProperties.DEFAULT_DOMAIN, ogConfigs);

ogManager.setOverrideObjectGridConfigurations(overrideMap);
ClientClusterContext client = ogManager.connect(catalogServerAddresses, null, null);
ObjectGrid companyGrid = ogManager.getObjectGrid(client, objectGridName);
```

Die ogManager-Instanz der Schnittstelle "ObjectGridManager" sucht nur nach Überschreibungen in den Objekten "ObjectGridConfiguration" und BackingMap-Configuration", die Sie in die overrideMap-Map einschließen. Der vorherige Code überschreibt beispielsweise die Anzahl der Buckets in der Map "OrderLine". Die Map "Order" bleibt jedoch auf der Clientseite unverändert, weil keine Konfiguration für diese Map eingefügt wurde.

Client in Spring Framework konfigurieren

Clientseitige ObjectGrid-Einstellungen können auch über Spring Framework überschrieben werden. Die folgende XML-Beispieldatei veranschaulicht, wie ein Element "ObjectGridConfiguration" erstellt und zum Überschreiben einige clientseitiger Einstellungen verwendet wird. Diese Beispieldatei ruft dieselben APIs auf, die auch in der Konfiguration über das Programm verwendet werden. Das Beispiel ist funktional äquivalent zu dem Beispiel in der ObjectGrid-XML-Konfiguration.

Clientkonfiguration mit Spring

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
  "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
  <bean id="companyGrid" factory-bean="manager" factory-method="getObjectGrid"
    singleton="true">
    <constructor-arg type="com.ibm.websphere.objectgrid.ClientClusterContext">
      <ref bean="client" />
    </constructor-arg>
    <constructor-arg type="java.lang.String" value="CompanyGrid" />
</bean>

  <bean id="manager" class="com.ibm.websphere.objectgrid.ObjectGridManagerFactory"
    factory-method="getObjectGridManager" singleton="true">
    <property name="overrideObjectGridConfigurations">
      <map>
        <entry key="DefaultDomain">
          <list>
            <ref bean="ogConfig" />
          </list>
        </entry>
      </map>
    </property>
  </bean>

  <bean id="ogConfig"
    class="com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory"
    factory-method="createObjectGridConfiguration">
    <constructor-arg type="java.lang.String">
      <value>CompanyGrid</value>
    </constructor-arg>
    <property name="plugins">
      <list>
   <bean class="com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory"
         factory-method="createPlugin">
         <constructor-arg type="com.ibm.websphere.objectgrid.config.PluginType"
           value="TRANSACTION_CALLBACK" />
         <constructor-arg type="java.lang.String"
           value="com.company.MyClientTxCallback" />
       </bean>
       <bean class="com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory"
         factory-method="createPlugin">
         <constructor-arg type="com.ibm.websphere.objectgrid.config.PluginType"
           value="OBJECTGRID_EVENT_LISTENER" />
         <constructor-arg type="java.lang.String" value="" />
       </bean>
```

```
     </list>
 </property>
     <property name="backingMapConfigurations">
       <list>
    <bean class="com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory"
           factory-method="createBackingMapConfiguration">
           <constructor-arg type="java.lang.String" value="Customer" />
             <property name="plugins">
     <bean class="com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory"
                 factory-method="createPlugin">
                 <constructor-arg type="com.ibm.websphere.objectgrid.config.PluginType"
    value="EVICTOR" />
   <constructor-arg type="java.lang.String"
                  value="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" />
               </bean>
             </property>
             <property name="numberOfBuckets" value="1429" />
           </bean>
           <bean class="com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory"
             factory-method="createBackingMapConfiguration">
               <constructor-arg type="java.lang.String" value="OrderLine" />
                 <property name="numberOfBuckets" value="701" />
   <property name="timeToLive" value="800" />
   <property name="ttlEvictorType">
                   <value type="com.ibm.websphere.objectgrid.
          TTLType">LAST_ACCESS_TIME</value>
   </property>
           </bean>
         </list>
       </property>
     </bean>

     <bean id="client" factory-bean="manager" factory-method="connect"
       singleton="true">
       <constructor-arg type="java.lang.String">
    <value>localhost:2809</value>
       </constructor-arg>
   <constructor-arg
          type="com.ibm.websphere.objectgrid.security.
     config.ClientSecurityConfiguration">
         <null />
       </constructor-arg>
   <constructor-arg type="java.net.URL">
     <null />
       </constructor-arg>
     </bean>
</beans>
```

Nachdem Sie die XML-Datei erstellt haben, laden Sie die Datei und erstellen das ObjectGrid mit dem folgenden Code-Snippet:

```
BeanFactory beanFactory = new XmlBeanFactory(new

UrlResource("file:test/companyGridSpring.xml"));

ObjectGrid companyGrid = (ObjectGrid) beanFactory.getBean("companyGrid");
```

Weitere Informationen zum Erstellen einer XML-Deskriptordatei finden Sie in der Übersicht über die Integration des Spring-Frameworks.

Nahen Clientcache inaktivieren

Der nahe Cache wird standardmäßig aktiviert, wenn eine optimistische Sperrstrategie oder keine Sperrstrategie konfiguriert ist. Clients verwalten keinen nahen Cache, wenn pessimistisches Sperren konfiguriert ist. Zum Inaktivieren des nahen Caches müssen Sie das Attribut "numberOfBuckets" in der ObjectGrid-Deskriptordatei für das Überschreiben des Clients auf 0 setzen.

Mechanismus für Clientinaktivierung aktivieren

In einer verteilten Umgebung von WebSphere eXtreme Scale gibt es auf der Clientseite standardmäßig einen nahen Cache, wenn die optimistische Sperrstrategie verwendet wird oder Sperren inaktiviert sind. Der nahe Cache enthält seine eigenen lokalen zwischengespeicherten Daten. Wenn ein Client von eXtreme Scale eine Aktualisierung festschreibt, wird diese Aktualisierung an den nahen Cache des Clients und an den Server gesendet. Andere Clients von eXtreme Scale erhalten die Aktualisierungsinformationen jedoch nicht und können daraufhin Daten haben, die nicht auf dem neuesten Stand sind.

Naher Cache

Anwendungen müssen sich dieses Problems potenziell veralteter Daten in Clients von eXtreme Scale bewusst sein. Sie können die integrierte JMS-basierte (Java Message Service) ObjectGridEventListener-Klasse "com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener" verwenden, um den Mechanismus für Clientinaktivierung in einer verteilten eXtreme-Scale-Umgebung, einem so genannten eXtreme-Scale-Grid, zu aktivieren.

Der Mechanismus für Clientinaktivierung ist die Lösung für das Problem veralteter Daten im nahen Cache des Clients in einer verteilten eXtreme-Scale-Umgebung. Diese Mechanismus stellt sicher, dass der nahe Cache des Clients mit Servern oder anderen Clients synchronisiert wird. Aber selbst mit diesem JMS-basierten Mechanismus für Clientinaktivierung wird der nahe Cache des Clients nicht sofort aktualisiert. Es tritt eine Verzögerung auf, wenn die Laufzeitumgebung von eXtreme Scale Aktualisierungen veröffentlicht.

Es sind zwei Modelle für den Mechanismus für Clientinaktivierung in einer verteilten eXtreme-Scale-Umgebung verfügbar:

- Client/Server-Modell: In diesem Modell haben alle Serverprozesse die Rolle "Publisher" (Bereitsteller), die alle Transaktionsänderungen an der vorgesehenen JMS-Destination veröffentlicht. Alle Clientprozesse haben die Rolle "Receiver" (Empfänger) und empfangen alle Transaktionsänderungen von der vorgesehenen JMS-Destination.
- Modell mit dem Client in zwei Rollen: In diesem Modell haben alle Serverprozesse nichts mit der JMS-Destination zu tun. Alle Clientprozesse übernehmen die Rollen "Publisher" (Veröffentlichung) und "Receiver" (Empfang) für die JMS-Destinations. Transaktionsänderungen, die auf Clientseite stattfinden, werden an der JMS-Destination veröffentlicht, und alle Clients empfangen diese Transaktionsänderungen.

Weitere Informationen finden Sie im Abschnitt „JMS-Ereignis-Listener” auf Seite 142.

Client/Server-Modell

In einem Client/Server-Modell haben die Server die Rolle "JMS-Publisher", und der Client hat die Rolle "JMS-Receiver".

XML-Beispiel für Client/Server-Modell

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="AgentObjectGrid">
      <bean id="ObjectGridEventListener"
        className="com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener">
        <property name="invalidationModel" type="java.lang.String" value="CLIENT_SERVER_MODEL" description="" />
```

```
        <property name="invalidationStrategy" type="java.lang.String" value="PUSH" description="" />
        <property name="mapsToPublish" type="java.lang.String" value="agent;profile;pessimisticMap" description="" />
        <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String" value="defaultTCF" description="" />
        <property name="jms_topicJndiName" type="java.lang.String" value="defaultTopic" description="" />
        <property name="jms_topicName" type="java.lang.String" value="defaultTopic" description="" />
        <property name="jms_userid" type="java.lang.String" value="" description="" />
        <property name="jms_password" type="java.lang.String" value="" description="" />
        <property name="jndi_properties" type="java.lang.String"
           value="java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory;
     java.naming.provider.url=
    tcp://localhost:61616;connectionFactoryNames=defaultTCF;topic.defaultTopic=defaultTopic"
          description="jndi properties" />
      </bean>

      <backingMap name="agent" readOnly="false" pluginCollectionRef="agent" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="28800" />
      <backingMap name="profile" readOnly="false" pluginCollectionRef="profile" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="pessimisticMap" readOnly="false" pluginCollectionRef="pessimisticMap" preloadMode="false"
        lockStrategy="PESSIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="excludedMap1" readOnly="false" pluginCollectionRef="excludedMap1" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="excludedMap2" readOnly="false" pluginCollectionRef="excludedMap2" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
    </objectGrid>
  </objectGrids>

  <backingMapPluginCollections>
    <backingMapPluginCollection id="agent">
      <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.AgentObjectTransformer" />
    </backingMapPluginCollection>
    <backingMapPluginCollection id="profile">
      <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.ProfileObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
        <property name="maxSize" type="int" value="2000" description="set max size for LRU evictor" />
        <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
        <property name="numberOfLRUQueues" type="int" value="50" description="set number of LRU queues" />
      </bean>
    </backingMapPluginCollection>

    <backingMapPluginCollection id="pessimisticMap" />
    <backingMapPluginCollection id="excludedMap1" />
    <backingMapPluginCollection id="excludedMap2" />
  </backingMapPluginCollections>

</objectGridConfig>
```

Modell mit dem Client in zwei Rollen

In diesem Modell übernimmt jeder Client sowohl die Rolle "JMS-Publisher" als auch die Rolle "JMS-Receiver". Der Client veröffentlicht alle festgeschriebenen Transaktionsänderungen an der vorgesehenen JMS-Destination und empfängt alle festgeschriebenen Transaktionsänderungen von anderen Clients. Der Server selbst hat in diesem Modell nichts mit JMS zu tun.

XML-Beispiel mit Zweirollenmodell

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="AgentObjectGrid">
      <bean id="ObjectGridEventListener"
        className="com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener">
        <property name="invalidationModel" type="java.lang.String" value="CLIENT_AS_DUAL_ROLES_MODEL" description="" />
        <property name="invalidationStrategy" type="java.lang.String" value="PUSH" description="" />
        <property name="mapsToPublish" type="java.lang.String" value="agent;profile;pessimisticMap" description="" />
        <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String" value="defaultTCF" description="" />
        <property name="jms_topicJndiName" type="java.lang.String" value="defaultTopic" description="" />
        <property name="jms_topicName" type="java.lang.String" value="defaultTopic" description="" />
        <property name="jms_userid" type="java.lang.String" value="" description="" />
        <property name="jms_password" type="java.lang.String" value="" description="" />
        <property name="jndi_properties" type="java.lang.String"
          value="java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory;java.naming.provider.url=
     tcp://localhost:61616;connectionFactoryNames=defaultTCF;topic.defaultTopic=defaultTopic"
          description="jndi properties" />
      </bean>

      <backingMap name="agent" readOnly="false" pluginCollectionRef="agent" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="28800" />
```

```
      <backingMap name="profile" readOnly="false" pluginCollectionRef="profile" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="pessimisticMap" readOnly="false" pluginCollectionRef="pessimisticMap" preloadMode="false"
        lockStrategy="PESSIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="excludedMap1" readOnly="false" pluginCollectionRef="excludedMap1" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="excludedMap2" readOnly="false" pluginCollectionRef="excludedMap2" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
    </objectGrid>
  </objectGrids>

  <backingMapPluginCollections>
    <backingMapPluginCollection id="agent">
      <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.AgentObjectTransformer" />
    </backingMapPluginCollection>
    <backingMapPluginCollection id="profile">
      <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.ProfileObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
        <property name="maxSize" type="int" value="2000" description="set max size for LRU evictor" />
        <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
        <property name="numberOfLRUQueues" type="int" value="50" description="set number of LRU queues" />
      </bean>
    </backingMapPluginCollection>

    <backingMapPluginCollection id="pessimisticMap" />
    <backingMapPluginCollection id="excludedMap1" />
    <backingMapPluginCollection id="excludedMap2" />
  </backingMapPluginCollections>

</objectGridConfig>
```

Zeitlimit für Anforderungswiederholung konfigurieren

Bei zuverlässigen Maps können Sie ein Wiederholungszeitlimit für Transaktionsanforderungen an WebSphere eXtreme Scale übergeben.

Es gibt zwei Methoden für die Konfiguration zuverlässiger Maps. Wenn ein Wert größer als null definiert wird, wird die Anforderung so lange wiederholt, bis das Zeitlimit abläuft oder ein permanenter Fehler wie eine Ausnahme des Typs "DuplicateKeyException" auftritt. Der Wert null steht für die Moduseinstellung "fail-fast", d. h., eXtreme Scale führt keine Wiederholung durch.

Sie geben einen Zeitlimitwert in Millisekunden in der Clienteigenschaftendatei oder in einer Sitzung an. Die Sitzungseinstellung überschreibt immer die Einstellung in der Clienteigenschaftendatei. Zur Laufzeit wird das Transaktionszeitlimit zusammen mit dem Wiederholungszeitlimit verwendet, um sicherzustellen, dass das Wiederholungszeitlimit nicht höher ist als das Transaktionszeitlimit.

Aufgrund der Variationen in der Beendigung von Transaktionen mit automatischer Festschreibung und ohne automatische Festschreibung (Transaktionen, die explizite Methoden "begin" und "commit" verwenden), sind auch die gültigen Ausnahmen für die Wiederholung verschieden.

Für Transaktionen, die innerhalb einer Sitzung aufgerufen werden, gilt die Wiederholung für CORBA-Ausnahmen des Typs "SystemExceptions" und eXtreme-Scale-Ausnahmen des Typs "TargetNotAvailable".

Für Transaktionen mit automatischer Festschreibung ist die Wiederholung für CORBA-Ausnahmen des Typs "SystemExceptions" und eXtreme-Scale-Ausnahmen gültig, die sich auf die Verfügbarkeit beziehen (ReplicationVotedToRollbackTransactionException, TargetNotAvailable, AvailabilityException usw.).

Weitere Informationen finden Sie im Abschnitt zur Verwendung von Sitzungen für den Zugriff auf Daten im Grid im *Programmierhandbuch*.

Anwendungs- oder andere permanente Fehler werden sofort zurückgegeben, und die Transaktion wird vom Client nicht wiederholt. Zu diesen permanenten Fehlern gehören Ausnahmen des Typs "DuplicateKeyException" und "KeyNotFoundException".

Bei der Einstellung für schnelles Fehlschlagen werden alle Ausnahmen ohne Wiederholung zurückgegeben.

Im Folgenden Sie die Ausnahmen ausführlicher beschrieben:

Ausnahmen mit Wiederholung auf Clientseite

- ReplicationVotedToRollbackTransactionException (nur bei automatischer Festschreibung)
- TargetNotAvailable
- org.omg.CORBA.SystemException
- AvailabilityException (nur bei automatischer Festschreibung)
- LockTimeoutException (nur bei automatischer Festschreibung)
- UnavailableServiceException (nur bei automatischer Festschreibung)

Weitere Ausnahmen

- DuplicateKeyException
- KeyNotFoundException
- LoaderException
- TransactionAffinityException
- LockDeadlockException
- OptimisticCollisionException

Eigenschaft "requestRetryTimeout" in einer Clienteigenschaftendatei festlegen

Zum Festlegen des requestRetryTimeout-Werts in einem Client müssen Sie in der „Clienteigenschaftendatei” auf Seite 207 die Eigenschaft "requestRetryTimeout" hinzufügen oder ändern. Die Clienteigenschaftendatei ist standardmäßig die Datei `objectGridClient.properties`. Die Eigenschaft "requestRetryTimeout" wird in Millisekunden festgelegt. Setzen Sie die Eigenschaft auf einen Wert größer als null, damit eine Anforderung bei Ausnahmen, für die eine Wiederholung verfügbar ist, wiederholt wird. Setzen Sie die Eigenschaft auf `0`, damit Anforderungen bei Ausnahmen ohne Wiederholung fehlschlagen. Zur Verwendung des Standardverhaltens entfernen Sie die Eigenschaft, oder setzen Sie sie auf den Wert `-1`.

objectGridClient.properties

```
# Konfiguration eines eXtreme-Scale-Clients
preferLocalProcess = false
preferLocalhost = false
requestRetryTimeout = 30000
```

Der requestRetryTimeout-Wert wird in Millisekunden angegeben. Wenn der Wert aus dem vorherigen Beispiel in einer ObjectGrid-Instanz verwendet wird, ist der requestRetryTimeout-Wert 30 Sekunden.

Clienteigenschaften durch programmgesteuerten Abruf einer ObjectGrid-Verbindung festlegen

Wenn Sie die Clienteigenschaften über das Programm festlegen möchten, müssen Sie zuerst eine Clienteigenschaftendatei an einer für Ihre Anwendung angemessenen Position erstellen. Im folgenden Beispiel verweist die Clienteigenschaftendatei auf das Snippet `objectGridClient.properties` im vorherigen Abschnitt. Nachdem Sie die Verbindung zum ObjectGridManager hergestellt haben, definieren Sie die Clienteigenschaften wie beschrieben. Die ObjectGrid-Instanz, die Sie erhalten, hat die Clienteigenschaften, die Sie in der Datei definiert haben. Wenn Sie die Clienteigenschaftendatei ändern, müssen Sie jedes Mal explizit eine neue ObjectGrid-Instanz anfordern.

```
ObjectGridManager manager = ObjectGridManagerFactory.getObjectGridManager();
String objectGridName = "testObjectGrid";
URL clientXML = null;
ClientClusterContext ccc = manager.connect("localhost:2809", null, clientXML);
File file = new File("<location>/objectGridClient.properties");
URL url = file.toURI().toURL();
ccc.setClientProperties(objectGridName, url);
ObjectGrid objectGrid = ogManager.getObjectGrid(ccc, objectGridName);
```

Beispiel für Sitzungsüberschreibung mit automatischer Festschreibung

Wenn Sie das Zeitlimit für Anforderungswiederholung in einer Sitzung festlegen oder die Clienteigenschaft "requestRetryTimeout" überschreiben möchten, rufen Sie die Methode "setRequestRetryTimeout(long)" in der Schnittstelle "Session" auf.

```
Session sessionA = objectGrid.getSession();
sessionA.setRequestRetryTimeout(30000);
ObjectMap mapA = sessionA.getMap("payroll");
String key = "key:" + j;
mapA.insert(key, "valueA");
```

Diese Sitzung verwendet jetzt einen requestRetryTimeout-Wert von 30000 Millisekunden bzw. 30 Sekunden, unabhängig von dem in der Clienteigenschaftendatei festgelegten Wert. Weitere Informationen zur Schnittstelle "Session" finden Sie im Abschnitt Sitzungen für den Zugriff auf Daten im Grid verwenden.

Entitäten konfigurieren

Ein ObjectGrid kann eine beliebige Anzahl logischer Entitätsschemas haben. Entitäten werden über annotierte Java-Klassen, XML oder eine Kombination von XML und Java-Klassen definiert. Definierte Entitäten werden anschließend bei einem eXtreme-Scale-Server registriert und an BackingMaps, Indizes und andere Plug-ins gebunden.

Vorbereitende Schritte

Ein Entitätsschema setzt sich aus einer Gruppe von Entitäten und den Beziehungen zwischen diesen Entitäten zusammen. Einzelheiten zur Schemadefinition und Entitätskonfiguration finden Sie unter Entitätsschema definieren.

Verwaltung von Beziehungen

Objektorientierte Sprachen wie Java und relationale Datenbanken unterstützen Beziehungen oder Assoziationen. Beziehungen verringern den Speicherbedarf durch Die Verwendung von Objektreferenzen und Fremdschlüsseln.

Wenn Sie Beziehungen in einem Grid verwenden, müssen die Daten in einer Baumstruktur mit Integritätsbedingungen organisiert werden. Es muss einen einzigen Stammtyp in der Baumstruktur geben, und alle untergeordneten Typen dürfen nur einem einzigen Stammtyp zugeordnet sein. Beispiel: Eine Abteilung kann viele Mitarbeiter und ein Mitarbeiter viele Projekte haben. Aber ein Projekt kann keine Mitarbeiter haben, die zu verschiedenen Abteilungen gehören. Nach der Definition eines Stammobjekts werden alle Zugriff auf dieses Stammobjekt und seine untergeordneten Objekte über das Stammobjekt verwaltet. WebSphere eXtreme Scale verwendet den Hash-Code des Stammobjektschlüssels, um eine Partition auszuwählen.

Beispiel: Partition = (Hash-Code MOD Anzahl_Partitionen)

Wenn alle Daten für eine Beziehung an eine einzige Objektinstanz gebunden sind, kann die gesamte Baumstruktur in einer einzigen Partition zusammengefasst werden, und der Zugriff auf diese Instanz kann sehr effizient über eine einzige Transaktion erfolgen. Wenn sich die Daten auf mehrere Beziehungen verteilen, müssen mehrere Partitionen beteiligt werden. Dies impliziert zusätzliche Fernaufrufe, was zu Leistungsengpässen führen kann.

Referenzdaten

Einige Beziehungen enthalten Such- oder Referenzdaten, wie z. B. "CountryName". Dies ist ein Sonderfall, in dem die Daten in jeder Partition vorhanden sein müssen. Hier kann der Zugriff auf die Daten über einen beliebigen Stammschlüssel erfolgen, und es wird immer dasselbe Ergebnis zurückgegeben. Referenzdaten wie diese sollten nur verwendet werden, wenn die Daten relativ statisch sind, da die Aktualisierung der Daten kostenintensiv sein kann, weil sie in jeder Partition durchgeführt werden muss. Die API "DataGrid" ist eine gängige Technik, mit der Referenzdaten auf dem aktuellen Stand gehalten werden können.

Kosten und Vorteile der Normalisierung

Durch die Normalisierung der Daten über Beziehungen kann der Speicherbedarf des Grids verringert werden, weil sich die Duplizierung der Daten verringert. Im Allgemeinen gilt jedoch, dass die horizontale Skalierung mit zunehmendem Volumen relationaler Daten abnimmt. Wenn Daten gruppiert werden, nimmt der Aufwand für die Verwaltung der Beziehungen und deren Größe zu. Da die Daten von Grid-Partitionen auf dem Schlüssel des Stammobjekts der Baumstruktur basieren, wird die Größe der Baumstruktur nicht berücksichtigt. Wenn Sie sehr viele Beziehungen für eine einzige Instanz der Baumstruktur haben, kann die Datenverteilung im Grid deshalb ungleichmäßig sein, d. h., eine Partition enthält mehr Daten als die anderen.

Wenn die Daten normalisiert oder reduziert werden, werden die Daten, die normalerweise von zwei Objekten gemeinsam genutzt werden, stattdessen dupliziert, und jede Tabelle kann gesondert partitioniert werden, wodurch eine gleichmäßigere Verteilung der Daten im Grid möglich ist. Dies erhöht zwar den Speicherbedarf, aber die Anwendung kann skaliert werden, da auf eine einzige Datenzeile zugegriffen werden kann, die alle erforderlichen Daten enthält. Dies ist ideal für die Grid, in denen hauptsächlich Leseoperationen durchgeführt werden, da die Verwaltung der Daten kostenintensiver wird.

Weitere Informationen finden Sie auf der Webseite "Classifying XTP systems and scaling".

Beziehungen über die Datenzugriffs-APIs verwalten

Die API "ObjectMap" ist die schnellste, flexibelste und differenzierteste der Datenzugriffs-APIs und unterstützt einen transaktionsorientierten, sitzungsbasierten Ansatz für den Zugriff auf Daten im Map-Grid. Die API "ObjectMap" ermöglicht Clients die Verwendung allgemeiner CRUD-Operationen (Create, Read, Update and Delete, Erstellen, Lesen, Aktualisieren und Löschen) für die Verwaltung von Schlüssel/Wert-Paaren für die Objekte im verteilten Grid.

Wenn Sie die API "ObjectMap" verwenden, müssen Objektbeziehungen durch Integration des Fremdschlüssels für alle Beziehungen im übergeordneten Objekt ausgedrückt werden.

Es folgt ein Beispiel:

```
public class Department {
 Collection<String> employeeIds;
}
```

Die API "EntityManager" vereinfacht die Verwaltung von Beziehungen, indem sie persistente Daten aus den Objekten extrahiert, einschließlich der Fremdschlüssel. Wenn das Objekt später aus dem Grid abgerufen wird, wird der Beziehungsgraph erneut erstellt, wie im folgenden Beispiel gezeigt wird:

```
@Entity
public class Department {
 Collection<String> employees;
}
```

Die API "EntityManager" ist anderen Java-Objektpersistenztechnologien wie JPA und Hibernate insofern sehr ähnlich, als sie einen Graph verwalteter Java-Objektinstanzen mit dem persistenten Speicher synchronisiert. In diesem Fall ist der persistente Speicher ein eXtreme-Scale-Grid, in dem jede Entität als Map dargestellt wird, die die Entitätsdaten und nicht die Objektinstanzen enthält.

XML-Deskriptordatei für Entitätsmetadaten

Die Deskriptordatei für Entitätsmetadaten ist eine XML-Datei, die verwendet wird, um ein Entitätsschema für WebSphere eXtreme Scale zu definieren. Definieren Sie alle Entitätsmetadaten in der XML-Datei, oder definieren Sie die Entitätsmetadaten als Annotationen in der Java-Klassendatei der Entität. Vorrangig wird die XML-Deskriptordatei für Entitäten verwendet, die keine Java-Annotationen verwenden können.

Verwenden Sie XML-Konfiguration, um Entitätsmetadaten zu erstellen, die auf der XML-Datei basieren. Wenn die XML-Konfiguration in Kombination mit Annotationen verwendet wird, überschreiben einige der Attribute, die in der XML-Konfiguration definiert werden, die entsprechenden Annotationen. Wenn Sie ein Element überschreiben, wird dies explizit in den folgenden Abschnitten hervorgehoben. Ein Beispiel für die XML-Deskriptordatei für Entitätsmetadaten finden Sie im Abschnitt „Datei `emd.xsd`" auf Seite 232.

Element "id"

Das Element "id" impliziert, dass das Attribut ein Schlüssel ist. Es muss mindestens ein Element "id" angegeben werden. Sie können mehrere id-Schlüssel als Verbundschlüssel angeben.

Attribute

name
Gibt den Namen des Attributs an. Das Attribut muss in der Java-Datei enthalten sein.

alias
Gibt den Elementalias an. Der Aliaswert wird überschrieben, wenn er zusammen mit einer annotierten Entität verwendet wird.

Element "basic"

Das Element "basic" impliziert, dass das Attribut ein primitiver Typ oder ein Wrapper für primitive Typen ist.

- java.lang.String
- java.math.BigInteger
- java.math.BigDecimal
- java.util.Date
- java.util.Calendar
- java.sql.Date
- java.sql.Time
- java.sql.Timestamp
- byte[]
- Byte[]
- char[]
- Character[]
- Java Platform, Standard Edition Version 5 enum

Es ist nicht erforderlich, Attribute für "basic" anzugeben. Die Attribute des Elements "basic" werden automatisch über Reflexion konfiguriert.

Element "id-class"

Das Element "id_class" gibt eine Verbundschlüsselklasse an, mit der Entitäten über Verbundschlüssel gesucht werden können.

Attribute

class-name
Gibt den Klassennamen (eine id-Klasse) an, der für das Element "id-class" verwendet werden soll.

transient

Das Element "transient" impliziert, dass es ignoriert und nicht verarbeitet wird. Es kann auch überschrieben werden, wenn es zusammen mit annotierten Entitäten verwendet wird.

Attribute

name
Gibt den Namen des Attributs an, das ignoriert wird.

version

Das Element "transient" impliziert, dass es ignoriert und nicht verarbeitet wird. Es kann auch überschrieben werden, wenn es zusammen mit annotierten Entitäten verwendet wird.

Attribute

name
Gibt den Namen des Attributs an, das ignoriert wird.

Element "property"

Verwenden Sie das Element "property", um Plug-ins Eigenschaften hinzuzufügen. Der Name der Eigenschaft muss einer set-Methode in der von der übergeordneten Bean referenzierten Klasse entsprechen.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

name
Der Name der Eigenschaft. Der Wert, der diesem Attribut zugeordnet wird, muss einer set-Methode in der Klasse entsprechen, die mit dem Attribut "className" der übergeordneten Bean angegeben wird. Wenn Sie beispielsweise das Attribut "className" der Bean auf `com.ibm.MyPlugin` setzen und der angegebene Name der Eigenschaft `size` lautet, muss die Klasse `com.ibm.MyPlugin` eine Methode "setSize" enthalten. (Erforderlich)

type
Gibt den Typ der Eigenschaft an. Der Typ wird an die set-Methode übergeben, die mit dem Attribut "name" angegeben wurde. Die gültigen Werte sind primitive Java-Typen, ihre java.lang-Pendants und java.lang.String. Die Attribute "name" und "type" müssen einer Methodensignatur im Attribut "className" der Bean entsprechen. Wenn Sie beispielsweise den Namen `size` und den Typ `int` festlegen, muss eine Methode "setSize(int)" in der Klasse vorhanden sein, die mit dem Attribut "className" für die Bean angegeben wurde. (Erforderlich)

value
Gibt den Wert der Eigenschaft an. Dieser Wert wird in den mit dem Attribut "type" angegebenen Typ konvertiert und anschließend als Parameter in dem Aufruf der set-Methode verwendet, die mit den Attributen "name" und "type" angegeben wurde. Der Wert dieses Attributs wird nicht validiert. (Erforderlich)

description
Beschreibt die Eigenschaft. (Optional)

```
<bean
(1)   name="name"
(2)   type="java.lang.String" | "boolean" | "java.lang.Boolean" | "int" |
       "java.lang.Integer" | "double" | "java.lang.Double" | "byte" |
       "java.lang.Byte" | "short" | "java.lang.Short" | "long" |
       "java.lang.Long" | "float" | "java.lang.Float" | "char" |
       "java.lang.Character"
(3)   value="value"
(4)   description="description"
/>
```

Im folgenden Beispiel wird die Datei companyGridProperty.xml verwendet, um zu veranschaulichen, wie einer Bean ein Element "property" hinzugefügt wird. In diesem Beispiel wird einem Evictor (Bereinigungsprogramm) eine Eigenschaft mit dem Namen "maxSize" und dem Typ "int" hinzugefügt. Der Evictor "com.ibm-.websphere.objectgrid.plugins.builtins.LRUEvictor" hat eine Methodensignatur, die der Methode "setMaxSize(int)" entspricht. Der ganzzahlige Wert 499 wird an die Methode "setMaxSize(int)" in der Klasse "com.ibm.websphere.objectgrid.plugins-.builtins.LRUEvictor" übergeben.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

 <objectGrids>
<objectGrid name="CompanyGrid">
<backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
<backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor>
     <property name="maxSize" type="int" value="449"
       description="The maximum size of the LRU Evictor"/>
   </bean>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei companyGridProperty.xml aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);

BackingMap customerMap = companyGrid.defineMap("Customer");

LRUEvictor lruEvictor = new com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor();
// Wenn Sie die XML-Datei verwenden würden, würde die hinzugefügte
// Eigenschaft den folgenden Aufruf bewirken
lruEvictor.setMaxSize(449);
customerMap.setEvictor(lruEvictor);
```

Element "backingMapPluginsCollections"

Das Element "backingMapPluginsCollections" ist ein Container für alle backing-MapPluginCollection-Elemente. In der Datei companyGridProperty.xml aus dem vorherigen Beispiel enthält das Element "backingMapPluginCollections" ein einziges Element "backingMapPluginCollection" mit der ID customerPlugins.

- Anzahl der Vorkommen: 0 bis 1
- Untergeordnetes Element: backingMapPluginCollection

Element "backingMapPluginCollection"

Das Element "backingMapPluginCollection" definiert die BackingMap-Plug-ins und wird über das Attribut **id** identifiziert. Geben Sie das Attribut "pluginCollection-Ref" an, um die Plug-ins zu referenzieren. Wenn Sie mehrere BackingMap-Plug-ins auf ähnliche Weise konfigurieren, kann jede BackingMap dasselbe Element "backingMapPluginCollection" referenzieren.

- Anzahl der Vorkommen: 0 bis viele

- Untergeordnetes Element: bean

Attribute

id Identifiziert die backingMapPluginCollection und wird vom Attribut "pluginCollectionRef" des Elements "backingMap" referenziert. Jede ID muss eindeutig sein. Wenn der Wert des Attributs "pluginCollectionRef" nicht mit der ID eines der backingMapPluginCollection-Elemente übereinstimmt, schlägt die XML-Validierung fehl. Es können beliebig viele backingMap-Elemente ein einziges Element "backingMapPluginCollection" referenzieren. (Erforderlich)

```
<backingMapPluginCollection
(1)   id="id"
/>
```

Im folgenden Beispiel wird die Datei `companyGridCollection.xml` verwendet, um zu veranschaulichen, wie das Element "backingMapPluginCollection" verwendet wird. In dieser Datei verwendet die BackingMap "Customer" die backingMapPluginCollection "customerPlugins", um die BackingMap "Customer" mit einem LRUEvictor-Plug-in zu konfigurieren. Die BackingMaps "Item" und "OrderLine" referenzieren die backingMapPluginCollection "collection2". Für diese BackingMaps ist jeweils ein LFUEvictor-Plug-in definiert.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

 <objectGrids>
<objectGrid name="CompanyGrid">
<backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
   <backingMap name="Item" pluginCollectionRef="collection2"/>
   <backingMap name="OrderLine"
    pluginCollectionRef="collection2"/>
<backingMap name="Order"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
<backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor/>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="collection2">
   <bean id="Evictor"
     className="com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor/>
   <bean id="OptimisticCallback"
     className="com.ibm.websphere.samples.objectgrid.EmployeeOptimisticCallBackImpl"/>
  </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridCollection.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();

ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
BackingMap customerMap = companyGrid .defineMap("Customer");
LRUEvictor customerEvictor = new LRUEvictor();
customerMap.setEvictor(customerEvictor);

BackingMap itemMap = companyGrid.defineMap("Item");
LFUEvictor itemEvictor = new LFUEvictor();
itemMap.setEvictor(itemEvictor);

BackingMap orderLineMap = companyGrid.defineMap("OrderLine");
LFUEvictor orderLineEvictor = new LFUEvictor();
orderLineMap.setEvictor(orderLineEvictor);

BackingMap orderMap = companyGrid.defineMap("Order");
```

Element "querySchema"

Das Element "querySchema" definiert Beziehungen zwischen BackingMaps und identifiziert den Objekttyp jeder Map. Diese Informationen werden von ObjectQuery verwendet, um Zeichenfolgen in der Abfragesprache in Map-Zugriffsaufrufe zu übersetzen. Weitere Informationen finden Sie in den Details zur Definition eines ObjectQuery-Schemas im *Programmierhandbuch*.

- Anzahl der Vorkommen: 0 bis 1
- Untergeordnete Elemente: mapSchemas, relationships

Element "mapSchemas"

Jedes Element "querySchema" hat ein einziges Element "mapSchemas", das ein oder mehrere mapSchema-Elemente enthält.

- Anzahl der Vorkommen: 1
- Untergeordnetes Element: mapSchema

Element "mapSchema"

Ein Element "mapSchema" definiert den Typ der Objekte, die in einer BackingMap gespeichert werden, und enthält Anweisungen zum Zugriff auf die Daten.

- Anzahl der Vorkommen: 1 oder mehr
- Untergeordnetes Element: Ohne

Attribute

mapName
Gibt den Namen der BackingMap an, die dem Schema hinzugefügt werden soll. (Erforderlich)

valueClass
Gibt den Typ des Objekts an, der im Wertabschnitt der BackingMap gespeichert wird. (Erforderlich)

primaryKeyField
Gibt den Namen des Primärschlüsselattributs im Attribut "valueClass" an. Der Primärschlüssel muss auch im Schlüsselabschnitt der BackingMap gespeichert werden. (Optional)

accessType
Gibt an, wie die Abfragesteuerkomponente die persistenten Daten in den valueClass-Objektinstanzen überwacht und auf diese zugreift. Wenn Sie das Attribut auf den Wert `FIELD` setzen, werden die Klassenfelder überwacht und dem Schema hinzugefügt. Wenn Sie das Attribut auf den Wert `PROPERTY` setzen, werden die Attribute, die den get- und is-Methoden zugeordnet sind, verwendet. Der Standardwert ist `PROPERTY`. (Optional)

```
<mapSchema
(1)   mapName="backingMapName"
(2)   valueClass="com.mycompany.OrderBean"
(3)   primaryKeyField="orderId"
(4)   accessType="PROPERTY" | "FIELD"
/>
```

Im folgenden Beispiel wird die Datei `companyGridQuerySchemaAttr.xml` verwendet, um eine Beispielkonfiguration für mapSchema zu veranschaulichen:

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

 <objectGrids>
<objectGrid name="CompanyGrid">
<backingMap name="Order"/>
   <backingMap name="Customer"/>

   <querySchema>
    <mapSchemas>
     <mapSchema mapName="Order"
      valueClass="com.mycompany.OrderBean"
      primaryKeyField="orderNumber"
      accessType="FIELD"/>
     <mapSchema mapName="Customer"
      valueClass="com.mycompany.CustomerBean"
      primaryKeyField="id"
      accessType="FIELD"/>
    </mapSchemas>
   </querySchema>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei companyGridQuerySchemaAttr.xml aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Schema definieren
QueryConfig queryCfg = new QueryConfig();
queryCfg.addQueryMapping(new QueryMapping(
    "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryMapping(new QueryMapping(
    "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
companyGrid.setQueryConfig(queryCfg);
```

Element "relationships"

Jedes Element "querySchema" hat kein oder ein Element "relationships", das eine oder mehrere Elemente "relationship" enthält.

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: relationship

Element "relationship"

Ein Element "relationship" definiert die Beziehung zwischen zwei BackingMaps sowie die Attribute im Attribut "valueClass", die für die Beziehungsbindung verwendet werden.

- Anzahl der Vorkommen: 1 oder mehr
- Untergeordnetes Element: Ohne

Attribute

source
Gibt den Namen der valueClass der Quellenseite einer Beziehung an. (Erforderlich)

target
Gibt den Namen der valueClass der Zielseite einer Beziehung an. (Erforderlich)

relationField
Gibt den Namen des Attributs in der Quellen-valueClass an, die auf das Ziel verweist. (Erforderlich)

invRelationField
Gibt den Namen des Attributs in der Ziel-valueClass an, die auf die Quelle verweist. Wenn Sie dieses Attribut nicht angeben, ist die Beziehung unidirektional. (Optional)

```
<mapSchema
(1)   source="com.mycompany.OrderBean"
(2)   target="com.mycompany.CustomerBean"
(3)   relationField="customer"
(4)   invRelationField="orders"
/>
```

Im folgenden Beispiel wird die Datei `companyGridQuerySchemaWithRelationshipAttr.xml` verwendet, um eine mapSchema-Musterkonfiguration zu veranschaulichen, die eine bidirektionale Beziehung enthält.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">

 <objectGrids>
<objectGrid name="CompanyGrid">
<backingMap name="Order"/>
   <backingMap name="Customer"/>

   <querySchema>
    <mapSchemas>
     <mapSchema mapName="Order"
      valueClass="com.mycompany.OrderBean"
      primaryKeyField="orderNumber"
      accessType="FIELD"/>
     <mapSchema mapName="Customer"
      valueClass="com.mycompany.CustomerBean"
      primaryKeyField="id"
      accessType="FIELD"/>
    </mapSchemas>
    <relationships>
     <relationship
      source="com.mycompany.OrderBean"
      target="com.mycompany.CustomerBean"
      relationField="customer"/>
      invRelationField="orders"/>
    </relationships>
   </querySchema>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridQuerySchemaWithRelationshipAttr.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Schema definieren
QueryConfig queryCfg = new QueryConfig();
queryCfg.addQueryMapping(new QueryMapping(
    "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryMapping(new QueryMapping(
```

```
    "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryRelationship(new QueryRelationship(
     OrderBean.class.getName(), CustomerBean.class.getName(), "customer", "orders"));
companyGrid.setQueryConfig(queryCfg);
```

Element "timeBasedDBUpdate"

Ein Element "timeBasedDBUpdate" definiert eine Konfiguration für eine zeitbasierte Datenbankaktualisierungskomponente. Ein Element "timeBasedDBUpdate" enthält Informationen dazu, wie oft neu eingefügte oder aktualisierte Datensätze aus der Datenbank mit Java Persistence API (JPA) abgerufen werden und wie die Daten in den entsprechenden ObjectGrid-Maps aktualisiert werden.

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: Ohne

Attribute

entityClass
Gibt den Namen der Entitätsklasse an, die für die Interaktion mit dem JPA-Provider verwendet wird. Der Entitätsklassenname wird verwendet, um JPA-Entitäten über Entitätsabfragen abzurufen. (Erforderlich)

persistenceUnitName
Gibt den Namen der JPA-Persistenzeinheit für die Erstellung einer JPA-Entitätsmanager-Factory an. Der Standardwert ist der Name der ersten in der Datei `persistence.xml` definierten Persistenzeinheit. (Optional)

mode
Gibt den Modus für die zeitbasierte Datenbankaktualisierung an. Der Standardmodus für die zeitbasierte Datenbankaktualisierung ist INVALIDATE_ONLY. Der Modus INVALIDATE_ONLY gibt an, dass die Einträge in der ObjectGrid-Map ungültig gemacht, wenn die entsprechenden Datensätze in der Datenbank geändert wurden. Der Modus UPDATE_ONLY gibt an, dass die vorhandenen Einträge in der ObjectGrid-Map mit den aktuellen Werten aus der Datenbank aktualisiert werden. Alle neu in die Datenbank eingefügten Datensätze werden jedoch ignoriert. Der Modus INSERT_UPDATE gibt an, dass die vorhandenen Einträge in der ObjectGrid-Map mit den aktuellen Werten aus der Datenbank aktualisiert werden. Außerdem werden alle neu in die Datenbank eingefügten Datensätze in die ObjectGrid-Map eingefügt. (Optional)

timestampField
Gibt den Namen des Zeitmarkenfelds an. Ein Zeitmarkenfeldwert wird verwendet, um die Zeit oder Folge der letzten Aktualisierung eines Datensatzes im Datenbank-Back-End anzugeben. (Optional)

jpaPropertyFactory
Gibt den Namen der JPAPropertyFactory-Implementierungsklasse bzw. Spring-Bean an. Die Schnittstelle "com.ibm.websphere.objectgrid.jpa.JPAPropertyFactory" wird verwendet, um eine Persistenzeigenschaftentabelle zu integrieren, die die JPA-Standardeigenschaften überschreibt. Verwenden Sie Beans des Spring-Frameworks, wenn zusätzliche Attribute in der JPAPropertyFactory-Instanz gesetzt werden müssen. Weitere Informationen finden Sie unter Übersicht über die Integration des Spring-Frameworks. (Optional)

```
<timeBasedDBUpdate
(1)   persistenceUnitName="SamplePU"
(2)   mode="INVALIDATE_ONLY" | "UPDATE_ONLY" | "INSERT_UPDATE"
(3)   timestampField="TIMESTAMP"
(4)   entityClass="entity class"
(5)   jpaPropertyFactory="JPA property factory class" | "{spring}bean name"
/>
```

Element "streamQuerySet"

Das Element "streamQuerySet" ist das Ausgangselement für die Definition eines Abfragesatzes für Datenströme.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnete Elemente: stream, view

Element "stream"

Das Element "stream" stellt einen Datenstrom für die Abfragesteuerkomponente für Datenströme dar. Jedes Attribut des Elements "stream" entspricht einer Methode in der Schnittstelle "StreamMetadata".

- Anzahl der Vorkommen: 1 bis viele
- Untergeordnetes Element: basic

Attribute

name
Gibt den Namen des Datenstroms an. Die Validierung schlägt fehl, wenn dieses Attribut nicht angegeben wird. (Erforderlich)

valueClass
Gibt den Klassentyp des Werts an, der in der ObjectMap des Datenstroms gespeichert wird. Der Klassentyp wird verwendet, um das Objekt in Datenstromereignisse zu konvertieren und um eine SQL-Anweisung zu generieren, wenn die Anweisung nicht angegeben ist. (Erforderlich)

sql
Gibt die SQL-Anweisung des Datenstroms an. Wenn Sie diese Eigenschaft nicht angeben, wird eine Datenstrom-SQL durch Reflexion der Attribute bzw. Zugriffsmethoden im Attribut "valueClass" bzw. durch Verwendung der Tupelattribute der Entitätsmetadaten generiert. (Optional)

access
Gibt den Typ für den Zugriff auf die Attribute der Wertklasse an. Wenn Sie das Attribut auf den Wert `FIELD` setzen, werden die Attribute durch Java-Reflexion direkt aus den Feldern abgerufen. Andernfalls werden Zugriffsmethoden für das Lesen der Attribute verwendet. Der Standardwert ist `PROPERTY`. (Optional)

```
<stream
(1)   name="streamName"
(2)   valueClass="streamMapClassType"
(3)   sql="streamSQL create stream stockQuote
       keyed by t ( transactionvolume INTEGER, price DECIMAL (9,2),
        issue VARCHAR(100) );"
(4)   access="PROPERTY" | "FIELD"
/>
```

Element "view"

Das Element "view" stellt eine Datenstromabfragesicht dar. Jedes Element "stream" entspricht einer Methode in der Schnittstelle "ViewMetadata".

- Anzahl der Vorkommen: 1 bis viele
- Untergeordnete Elemente: basic, id

Attribute

name
Gibt den Namen der Sicht an. Die Validierung schlägt fehl, wenn dieses Attribut nicht angegeben wird. (Erforderlich)

sql
Gibt die SQL des Datenstroms an, die die Sichtkonvertierung definiert. Die Validierung schlägt fehl, wenn dieses Attribut nicht angegeben wird. (Erforderlich)

valueClass
Gibt den Klassentyp des Werts an, der in dieser Sicht der ObjectMap gespeichert wird. Der Klassentyp wird verwendet, um die Sichtereignisse in das richtige Tupelformat zu konvertieren, das mit diesem Klassentyp kompatibel ist. Wenn Sie den Klassentyp nicht angeben, wird ein Standardformat gemäß den Spaltendefinition in Stream Processing Technology Structured Query Language (SPTSQL) verwendet. Wenn Entitätsmetadaten für diese Sicht-Map definiert sind, darf dieses Attribut nicht verwendet werden. Stattdessen werden die Entitätsmetadaten verwendet. (Optional)

access
Gibt den Typ für den Zugriff auf die Attribute der Wertklasse an. Wenn Sie den Zugriffstyp auf `FIELD` setzen, werden die Spaltenwerte durch Java-Reflexion direkt aus den Feldern abgerufen. Andernfalls werden Zugriffsmethoden für das Definieren der Attribute verwendet. Der Standardwert ist `PROPERTY`. (Optional)

```
<view
(1)   name="viewName"
(2)   valueClass="viewMapValueClass"
(3)   sql="viewSQL CREATE VIEW last5MinuteAvgPrice AS
      SELECT issue, avg(price) as totalVolume
        FROM (SELECT * FROM stockQuote FETCH LATEST 5 MINUTES) group by issue;"/>
(4)   access="PROPERTY" | "FIELD"
/>
```

Element "basic"

Das Element "basic" wird verwendet, um eine Zuordnung des Attributnamens in der Wertklasse bzw. den Entitätsmetadaten zu der in SPTSQL definierten Spalte zu definieren.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

```
<basic
(1)   name="attributeName"
(2)   column="columnName"
/>
```

Element "id"

Das Element "id" wird für die Zuordnung eines Schlüsselattributs verwendet.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

```
<id
(1)   name="idName"
(2)   column="columnName"
/>
```

Im folgenden Beispiel wird die Datei `StreamQueryApp2.xml` verwendet, um zu veranschaulichen, wie die Attribut eines Datenstromabfragesatzes konfiguriert werden. Der Datenstromabfragesatz "_stockQuoteSQS_" hat einen Datenstrom und eine Sicht. Datenstrom und Sicht definieren einen Namen, die Werteklasse, die SQL und den Zugriffstyp. Der Datenstrom definiert auch ein Element "basic", das angibt, dass das Attribut "volume" in der Klasse "StockQuote" der in der SQL-Anweisung definierten SQL-Spalte "transactionvolume" zugeordnet ist.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
 <objectGrids>
  <objectGrid name="og1">
   <backingMap name="stockQuote" readOnly="false" copyKey="true"
    streamRef="stockQuote"/>
   <backingMap name="last5MinuteAvgPrice" readOnly="false" copyKey="false"
    viewRef="last5MinuteAvgPrice"/>

   <streamQuerySet name="stockQuoteSQS">
    <stream
     name="stockQuote"
     valueClass="com.ibm.ws.objectgrid.streamquery.sample.guide.StockQuote"
     sql="create stream stockQuote
         keyed by t ( transactionvolume INTEGER, price DECIMAL (9,2),
        issue VARCHAR(100) );"
     access="FIELD">
     <basic name="volume" column="transactionvolume"/>
    </stream>

    <view
     name="last5MinuteAvgPrice"
     valueClass="com.ibm.ws.objectgrid.streamquery.sample.guide.AveragePrice"
     sql="CREATE VIEW last5MinuteAvgPrice AS SELECT issue, avg(price) as avgPrice
       FROM (SELECT * FROM stockQuote FETCH LATEST 5 MINUTES) group by issue;"
     access="FIELD"
    </view>
   </streamQuerySet>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Datei `emd.xsd`

Verwenden Sie die XML-Schemadefinition für Entitätsmetadaten, um eine XML-Deskriptordatei zu erstellen und ein Entitätsschema für WebSphere eXtreme Scale zu definieren.

Beschreibungen der einzelnen Elemente und Attribute in der Datei `emd.xsd` finden Sie im Abschnitt „XML-Deskriptordatei für Entitätsmetadaten” auf Seite 221.

Datei `emd.xsd`

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:emd="http://ibm.com/ws/projector/config/emd"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     targetNamespace="http://ibm.com/ws/projector/config/emd"
     elementFormDefault="qualified" attributeFormDefault="unqualified"
    version="1.0">

    <!-- ************************************************* -->
    <xsd:element name="entity-mappings">
        <xsd:complexType>
            <xsd:sequence>
                   <xsd:element name="description" type="xsd:string" minOccurs="0"/>
                    <xsd:element name="entity" type="emd:entity" minOccurs="1" maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
          <xsd:unique name="uniqueEntityClassName">
            <xsd:selector xpath="emd:entity" />
```

```
            <xsd:field xpath="@class-name"/>
        </xsd:unique>
</xsd:element>

<!-- ************************************************** -->
<xsd:complexType name="entity">
    <xsd:sequence>
            <xsd:element name="description" type="xsd:string" minOccurs="0"/>
            <xsd:element name="id-class" type="emd:id-class" minOccurs="0"/>
            <xsd:element name="attributes" type="emd:attributes" minOccurs="0"/>
            <xsd:element name="entity-listeners" type="emd:entity-listeners" minOccurs="0"/>
            <xsd:element name="pre-persist" type="emd:pre-persist" minOccurs="0"/>
            <xsd:element name="post-persist" type="emd:post-persist" minOccurs="0"/>
            <xsd:element name="pre-remove" type="emd:pre-remove" minOccurs="0"/>
            <xsd:element name="post-remove" type="emd:post-remove" minOccurs="0"/>
            <xsd:element name="pre-invalidate" type="emd:pre-invalidate" minOccurs="0"/>
            <xsd:element name="post-invalidate" type="emd:post-invalidate" minOccurs="0"/>
            <xsd:element name="pre-update" type="emd:pre-update" minOccurs="0"/>
            <xsd:element name="post-update" type="emd:post-update" minOccurs="0"/>
            <xsd:element name="post-load" type="emd:post-load" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="class-name" type="xsd:string" use="required"/>
      <xsd:attribute name="access" type="emd:access-type"/>
      <xsd:attribute name="schemaRoot" type="xsd:boolean"/>
</xsd:complexType>

<!-- ************************************************** -->
 <xsd:complexType name="attributes">
    <xsd:sequence>
            <xsd:choice>
                 <xsd:element name="id" type="emd:id" minOccurs="0" maxOccurs="unbounded"/>
            </xsd:choice>
            <xsd:element name="basic" type="emd:basic" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="version" type="emd:version" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="many-to-one" type="emd:many-to-one" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="one-to-many" type="emd:one-to-many" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="one-to-one" type="emd:one-to-one" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element name="many-to-many" type="emd:many-to-many" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="transient" type="emd:transient" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
</xsd:complexType>

<!-- ************************************************** -->
 <xsd:simpleType name="access-type">
      <xsd:restriction base="xsd:token">
            <xsd:enumeration value="PROPERTY"/>
            <xsd:enumeration value="FIELD"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- ************************************************** -->
 <xsd:complexType name="id-class">
      <xsd:attribute name="class-name" type="xsd:string" use="required"/>
</xsd:complexType>

<!-- ************************************************** -->
<xsd:complexType name="id">
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="type" type="xsd:string" />
      <xsd:attribute name="alias" type="xsd:string" use="optional"/>
</xsd:complexType>

<!-- ************************************************** -->
 <xsd:complexType name="transient">
    <xsd:attribute name="name" type="xsd:string" use="required" />
</xsd:complexType>

<!-- ************************************************** -->
<xsd:complexType name="basic">
    <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="alias" type="xsd:string"/>
    <xsd:attribute name="type" type="xsd:string" />
      <xsd:attribute name="fetch" type="emd:fetch-type"/>
</xsd:complexType>

<!-- ************************************************** -->
 <xsd:simpleType name="fetch-type">
      <xsd:restriction base="xsd:token">
            <xsd:enumeration value="LAZY"/>
            <xsd:enumeration value="EAGER"/>
    </xsd:restriction>
</xsd:simpleType>

<!-- ************************************************** -->
 <xsd:complexType name="many-to-one">
    <xsd:sequence>
            <xsd:element name="cascade" type="emd:cascade-type" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="alias" type="xsd:string"/>
```

```
      <xsd:attribute name="target-entity" type="xsd:string"/>
      <xsd:attribute name="fetch" type="emd:fetch-type"/>
      <xsd:attribute name="id" type="xsd:boolean"/>
</xsd:complexType>
<!-- ************************************************** -->
 <xsd:complexType name="one-to-one">
    <xsd:sequence>
          <xsd:element name="cascade" type="emd:cascade-type" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="alias" type="xsd:string"/>
      <xsd:attribute name="target-entity" type="xsd:string"/>
      <xsd:attribute name="fetch" type="emd:fetch-type"/>
      <xsd:attribute name="mapped-by" type="xsd:string"/>
      <xsd:attribute name="id" type="xsd:boolean"/>
</xsd:complexType>
<!-- ************************************************** -->
 <xsd:complexType name="one-to-many">
    <xsd:sequence>
          <xsd:element name="order-by" type="emd:order-by" minOccurs="0"/>
          <xsd:element name="cascade" type="emd:cascade-type" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="alias" type="xsd:string"/>
      <xsd:attribute name="target-entity" type="xsd:string"/>
      <xsd:attribute name="fetch" type="emd:fetch-type"/>
      <xsd:attribute name="mapped-by" type="xsd:string"/>
</xsd:complexType>

<!-- ************************************************** -->
 <xsd:complexType name="many-to-many">
    <xsd:sequence>
          <xsd:element name="order-by" type="emd:order-by" minOccurs="0"/>
          <xsd:element name="cascade" type="emd:cascade-type" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
      <xsd:attribute name="alias" type="xsd:string"/>
      <xsd:attribute name="target-entity" type="xsd:string"/>
      <xsd:attribute name="fetch" type="emd:fetch-type"/>
      <xsd:attribute name="mapped-by" type="xsd:string"/>
</xsd:complexType>

<!-- ************************************************** -->
 <xsd:simpleType name="order-by">
      <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<!-- ************************************************** -->
 <xsd:complexType name="cascade-type">
    <xsd:sequence>
          <xsd:element name="cascade-all" type="emd:emptyType" minOccurs="0"/>
          <xsd:element name="cascade-persist" type="emd:emptyType" minOccurs="0"/>
          <xsd:element name="cascade-remove" type="emd:emptyType" minOccurs="0"/>
          <xsd:element name="cascade-invalidate" type="emd:emptyType" minOccurs="0"/>
          <xsd:element name="cascade-merge" type="emd:emptyType" minOccurs="0"/>
          <xsd:element name="cascade-refresh" type="emd:emptyType" minOccurs="0"/>
    </xsd:sequence>
</xsd:complexType>

<!-- ************************************************** -->
<xsd:complexType name="emptyType" />

<!-- ************************************************** -->
 <xsd:complexType name="version">
        <xsd:attribute name="name" type="xsd:string" use="required"/>
        <xsd:attribute name="alias" type="xsd:string"/>
      <xsd:attribute name="type" type="xsd:string" />
</xsd:complexType>

<!-- ************************************************** -->

 <xsd:complexType name="entity-listeners">
    <xsd:sequence>
          <xsd:element name="entity-listener" type="emd:entity-listener" minOccurs="0" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<!-- ************************************************** -->
 <xsd:complexType name="entity-listener">
    <xsd:sequence>
          <xsd:element name="pre-persist" type="emd:pre-persist" minOccurs="0"/>
          <xsd:element name="post-persist" type="emd:post-persist" minOccurs="0"/>
          <xsd:element name="pre-remove" type="emd:pre-remove" minOccurs="0"/>
          <xsd:element name="post-remove" type="emd:post-remove" minOccurs="0"/>
          <xsd:element name="pre-invalidate" type="emd:pre-invalidate" minOccurs="0"/>
          <xsd:element name="post-invalidate" type="emd:post-invalidate" minOccurs="0"/>
          <xsd:element name="pre-update" type="emd:pre-update" minOccurs="0"/>
          <xsd:element name="post-update" type="emd:post-update" minOccurs="0"/>
          <xsd:element name="post-load" type="emd:post-load" minOccurs="0"/>
    </xsd:sequence>
      <xsd:attribute name="class-name" type="xsd:string" use="required"/>
```

```
    </xsd:complexType>

    <!-- ******************************************************* -->
     <xsd:complexType name="pre-persist">
          <xsd:attribute name="method-name" type="xsd:string" use="required"/>
    </xsd:complexType>

    <!-- ******************************************************* -->
     <xsd:complexType name="post-persist">
          <xsd:attribute name="method-name" type="xsd:string" use="required"/>
    </xsd:complexType>

    <!-- ******************************************************* -->
    <xsd:complexType name="pre-remove">
          <xsd:attribute name="method-name" type="xsd:string" use="required"/>
    </xsd:complexType>

    <!-- ******************************************************* -->
     <xsd:complexType name="post-remove">
          <xsd:attribute name="method-name" type="xsd:string" use="required"/>
    </xsd:complexType>

    <!-- ******************************************************* -->
    <xsd:complexType name="pre-invalidate">
          <xsd:attribute name="method-name" type="xsd:string" use="required"/>
    </xsd:complexType>

    <!-- ******************************************************* -->
    <xsd:complexType name="post-invalidate">
          <xsd:attribute name="method-name" type="xsd:string" use="required"/>
    </xsd:complexType>

    <!-- ******************************************************* -->
     <xsd:complexType name="pre-update">
          <xsd:attribute name="method-name" type="xsd:string" use="required"/>
    </xsd:complexType>

    <!-- ******************************************************* -->
     <xsd:complexType name="post-update">
          <xsd:attribute name="method-name" type="xsd:string" use="required"/>
    </xsd:complexType>

    <!-- ******************************************************* -->
     <xsd:complexType name="post-load">
          <xsd:attribute name="method-name" type="xsd:string" use="required"/>
    </xsd:complexType>

</xsd:schema>
```

Cacheintegration konfigurieren

WebSphere eXtreme Scale kann in andere Caching-Produkte integriert werden. JPA kann zwischen WebSphere eXtreme Scale und der Datenbank verwendet werden, um Änderungen als Loader zu integrieren. Sie können auch den dynamischen Cacheprovider von WebSphere eXtreme Scale verwenden, um WebSphere eXtreme Scale als Plug-in in der dynamischen Cachekomponente von WebSphere Application Server zu verwenden. Eine andere Erweiterung von WebSphere Application Server ist der HTTP-Sitzungsmanager von WebSphere eXtreme Scale, der als Unterstützung für die Zwischenspeicherung von HTTP-Sitzungen eingesetzt werden kann.

Übersicht über die Cacheintegration: JPA, Sitzungen und dynamisches Caching

Das entscheidende Element, das WebSphere eXtreme Scale eine solche Vielseitigkeit und Zuverlässigkeit ermöglicht, ist die Anwendung von Caching-Konzepten für die Optimierung der Persistenz und erneuten Erfassung von Daten in praktisch jeder Implementierungsumgebung.

JPA-Loader konfigurieren

Ein JPA-Loader (Java Persistence API (JPA)) ist eine Plug-in-Implementierung, die JPA für die Interaktion mit der Datenbank verwendet.

Vorbereitende Schritte

- Sie müssen eine JPA-Implementierung wie Hibernate oder OpenJPA haben.
- Als Datenbank kann jedes Back-End verwendet werden, das vom ausgewählten JPA-Provider unterstützt wird.
- Sie können das JPALoader-Plug-in verwenden, wenn Sie Daten über die API "ObjectMap" speichern. Verwenden Sie das JPAEntityLoader-Plug-in, wenn Sie Daten über die API "EntityManager" speichern.

Informationen zu diesem Vorgang

Weitere Informationen zur Funktionsweise des JPA-Loaders finden Sie in den Informationen in der *Produktübersicht*.

Vorgehensweise

1. Konfigurieren Sie die Parameter, die JPA erfordert, um mit einer Datenbank zu interagieren.

   Die folgenden Parameter sind erforderlich. Diese Parameter werden in der Bean "JPALoader" oder "JPAEntityLoader" und der Bean "JPATxCallback" konfiguriert.

   - **persistenceUnitName**: Gibt den Namen der Persistenzeinheit an. Dieser Parameter wird für zwei Zwecke benötigt: zum Erstellen einer JPA-EntityManagerFactory und für das Suchen der JPA-Entitätsmetadaten in der Datei `persistence.xml`. Dieses Attribut wird in der Bean "JPATxCallback" gesetzt.
   - **JPAPropertyFactory**: Gibt die Factory zum Erstellen einer Map für Persistenzeigenschaften an, mit denen die Standardpersistenzeigenschaften überschrieben werden sollen. Dieses Attribut wird in der Bean "JPATxCallback" gesetzt. Um dieses Attribut zu setzen, ist eine Spring-Konfiguration erforderlich.
   - **entityClassName**: Gibt den Namen der Entitätsklasse an, die erforderlich ist, um JPA-Methoden wie EntityManager.persist, EntityManager.find usw. zu verwenden. Der JPALoader erfordert diesen Parameter, aber der Parameter ist für **JPAEntityLoader** optional. Wenn der Parameter **entityClassName** für einen JPAEntityLoader nicht konfiguriert ist, wird die in der ObjectGrid-Entitäts-Map konfigurierte Entitätsklasse verwendet. Sie müssen für den EntityManager von eXtreme Scale und den JPA-Provider dieselbe Klasse verwenden. Dieses Attribut wird in der Bean "JPALoader" oder "JPAEntityLoader" gesetzt.
   - **preloadPartition**: Gibt die Partition an, bei der der Preload-Prozess für die Map gestartet wird. Wenn die Preload-Partitionsnummer kleiner als null oder größer als die Gesamtanzahl der Partitionen minus 1 ist, wird der Preload-Prozess für die Map nicht gestartet. Der Standardwert ist -1 und bedeutet, dass der Preload-Prozess standardmäßig nicht gestartet wird. Dieses Attribut wird in der Bean "JPALoader" oder "JPAEntityLoader" gesetzt.

   Neben den JPA-Parametern, die in eXtreme Scale konfiguriert werden müssen, werden JPA-Metadaten verwendet, um den Schlüssel von den JPA-Entitäten abzurufen. Die JPA-Metadaten können als Annotation oder in einer Datei `orm.xml` konfiguriert werden, die in der Datei `persistence.xml` angegeben wird. Sie sind nicht Teil der Konfiguration von eXtreme Scale.
2. Konfigurieren Sie XML-Dateien für die JPA-Konfiguration.

   Informationen zum Konfigurieren eines JPALoader oder JPAEntityLoader finden Sie in den Beschreibungen von Loader-Plug-ins im *Programmierhandbuch*.

Konfigurieren Sie ein JPATxCallback-Transaktions-Callback zusammen mit der Loader-Konfiguration. Das folgende Beispiel ist eine ObjectGrid-XML-Deskriptordatei (objectgrid.xml), in der ein JPAEntityLoader und ein JPATxCallback konfiguriert sind:

Loader mit Callback konfigurieren - XML-Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
      <objectGrid name="JPAEM" entityMetadataXMLFile="jpaEMD.xml">
        <bean id="TransactionCallback"
           className="com.ibm.websphere.objectgrid.jpa.JPATxCallback">
           <property
              name="persistenceUnitName"
              type="java.lang.String"
              value="employeeEMPU" />
       </bean>
       <backingMap name="Employee" pluginCollectionRef="Employee" />
    </objectGrid>
  </objectGrids>

  <backingMapPluginCollections>
    <backingMapPluginCollection id="Employee">
        <bean id="Loader"
          className="com.ibm.websphere.objectgrid.jpa.JPAEntityLoader">
       <property
               name="entityClassName"
               type="java.lang.String"
               value="com.ibm.ws.objectgrid.jpa.test.entity.Employee"/>
        </bean>
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Wenn Sie eine JPAPropertyFactory konfigurieren möchten, müssen Sie eine Spring-Konfiguration verwenden. Im Folgenden sehen Sie eine XML-Musterkonfigurationsdatei mit dem Namen JPAEM_spring.xml, in der eine Spring-Bean für die Konfigurationen von eXtreme Scale konfiguriert wird.

Loader mit JPA-Eigenschaften-Factory konfigurieren - XML-Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

  <objectgrid:JPAEntityLoader id="jpaLoader"
  entityClassName="com.ibm.ws.objectgrid.jpa.test.entity.Employee"/>
  <objectgrid:JPATxCallback id="jpaTxCallback" persistenceUnitName="employeeEMPU" />
</beans>
```

Die XML-Konfigurationsdatei Objectgrid.xml folgt. Beachten Sie, dass der ObjectGrid-Name JPAEM ist und dem ObjectGrid-Namen in der Spring-Konfigurationsdatei JPAEM_spring.xml entspricht.

JPAEM-Loader-Konfiguration - XML-Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="JPAEM" entityMetadataXMLFile="jpaEMD.xml">
      <bean id="TransactionCallback"
            className="{spring}jpaTxCallback"/>
        <backingMap name="Employee" pluginCollectionRef="Employee"
                    writeBehind="T4"/>
      </objectGrid>
  </objectGrids>
```

```
  <backingMapPluginCollections>
    <backingMapPluginCollection id="Employee">
       <bean id="Loader" className="{spring}jpaLoader" />
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Eine Entität kann mit den JPA-Annotationen und mit den EntityManager-Annotationen von eXtreme Scale annotiert werden. Jede Annotation hat ein funktional entsprechendes XML-Element, das verwendet werden kann. Deshalb wurde in eXtreme Scale der Spring-Namespace hinzugefügt. Sie können diese Annotationen auch über die Spring-Namespace-Unterstützung konfigurieren.

Zeitbasierte JPA-Aktualisierungskomponente konfigurieren

Sie können eine zeitbasierte Datenbankaktualisierung mit XML für eine lokale oder verteilte Konfiguration von eXtreme Scale konfigurieren. Eine lokale Konfiguration kann auch programmgesteuert konfiguriert werden.

Informationen zu diesem Vorgang

Weitere Informationen zur Funktionsweise der zeitbasierten JPA-Datenaktualisierungskomponente finden Sie in den Informationen im *Programmierhandbuch*.

Vorgehensweise

Erstellen Sie eine timeBasedDBUpdate-Konfiguration.

- **Mit einer XML-Datei:**

  Das folgende Beispiel zeigt eine Datei `objectgrid.xml`, die eine timeBasedDBUpdate-Konfiguration enthält:

  Zeitbasierte JPA-Aktualisierungskomponente - XML-Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
        <objectGrid name="changeOG"
            entityMetadataXMLFile="userEMD.xml">
            <backingMap name="user" >
                <timeBasedDBUpdate timestampField="rowChgTs"
                      persistenceUnitName="userderby"
                      entityClass="com.test.UserClass"
                      mode="INVALIDATE_ONLY"
                />
            </backingMap>
        </objectGrid>
    </objectGrids>
    <backingMapPluginCollections>
</objectGridConfig>
```

  In diesem Beispiel wird die Map "user" mit einer zeitbasierten Datenbankaktualisierung konfiguriert. Der Datenbankaktualisierungsmodus ist INVALIDATE_ONLY, und das Zeitmarkenfeld ist "rowChgTs".

  Wenn das verteilte ObjectGrid "changeOG" im Containerserver gestartet wird, wird automatisch ein Thread für die zeitbasierte Datenbankaktualisierung in Partition 0 gestartet.

- **Programmgesteuert:**

  Wenn Sie ein lokales ObjectGrid erstellen, können Sie auch ein TimeBasedDBUpdateConfig-Objekt erstellen und in der BackingMap-Instanz definieren:

```
public void setTimeBasedDBUpdateConfig(TimeBasedDBUpdateConfig dbUpdateConfig);
```

  Weitere Informationen zum Definieren eines Objekts in der BackingMap-Instanz finden Sie in den Informationen zur Schnittstelle "BackingMap" in der API-Dokumentation.

Alternativ können Sie das Zeitmarkenfeld in der Entitätsklasse mit der Annotation "com.ibm.websphere.objectgrid.jpa.dbupdate.annotation.Timestamp" annotieren. Wenn Sie den Wert in der Klasse konfigurieren, müssen Sie das Zeitmarkenfeld nicht in der XML-Konfiguration konfigurieren.

Nächste Schritte

Starten Sie die zeitbasierte JPA-Datenaktualisierungskomponente. Weitere Informationen finden Sie in den Informationen zum Starten der zeitbasierten JAP-Datenaktualisierungskomponente im *Programmierhandbuch*.

JPA-Cache-Plug-ins konfigurieren

WebSphere eXtreme Scale enthält Cache-Plug-ins der Stufe 2 für die JPA-Provider OpenJPA und Hibernate.

Konfigurationseigenschaften des ObjectGrid-JPA-Caches

Sie können das JPA-Cache-Plug-in mit den folgenden Eigenschaften konfigurieren, die alle optional sind.

ObjectGridName
Gibt den eindeutigen ObjectGrid-Namen an. Der Standardwert ist der Name der definierten Persistenzeinheit. Wenn der Name der Persistenzeinheit nicht über den JPA-Provider verfügbar ist, wird ein generierter Name verwendet.

ObjectGridType
Gibt den Typ des ObjectGrids an.

Gültige Werte:

- **EMBEDDED**: Der Standard- und empfohlene Konfigurationstyp. Zu den Standardeinstellungen gehören `NumberOfPartitions=1`, `ReplicaMode=SYNC`, `ReplicaReadEnabled=true` und `MaxNumberOfReplicas=47`. Verwenden Sie den Parameter **ReplicaMode**, um den Replikationsmodus festzulegen, und den Parameter **MaxNumberOfReplicas**, um die maximale Anzahl an Replikaten festzulegen. Wenn ein System mehr als 47 Java Virtual Machines hat, setzen Sie **MaxNumberOfReplicas** auf die Anzahl der Java Virtual Machines.
- **EMBEDDED_PARTITION**: Der zu verwendende Typ, wenn das System hohe Datenvolumen in einem verteilten System zwischenspeichern muss. Die Standardanzahl an Partitionen ist `47` beim Replikationsmodus `NONE`. In einem kleinen System, das nur wenige Java Virtual Machines hat, setzen Sie **NumberOfPartitions** auf einen Wert kleiner-gleich der Anzahl Java Virtual Machines. Sie können Werte für **ReplicaMode**, **NumberOfPartitions** und **ReplicaReadEnabled** angeben, um das System zu optimieren.
- **REMOTE**: Der Cache versucht, über den Katalogservice eine Verbindung zu einem fernen, verteilten ObjectGrid herzustellen.

NumberOfPartitions

Gültige Werte: Größer-gleich 1Gibt die Anzahl der für den Cache zu verwendenden Partitionen an. Diese Eigenschaft gilt, wenn `EMBEDDED_PARTITION` als Wert für ObjectGridType angegeben ist. Der Standardwert ist `47`. Für den Typ `EMBEDDED` ist der Wert von **NumberOfPartitions** immer 1.

ReplicaMode

Gültige Werte: SYNC/ASYNC/NONEGibt die Methode an, die verwendet wird, um den Cache in die Replikate zu kopieren. Diese Eigenschaft gilt, wenn Sie EMBEDDED oder EMBEDDED_PARTITION als Wert für ObjectGridType festgelegt haben. Der Standardwert ist NONE für den Typ EMBEDDED_PARTITION und SYNC für den Typ EMBEDDED. Wenn Sie **ReplicaMode** auf NONE und EMBEDDED für ObjectGridType angeben, verwendet der Typ EMBEDDED weiterhin den **ReplicaMode**-Wert SYNC.

ReplicaReadEnabled

Gültige Werte: TRUE oder FALSEWenn Sie diese Eigenschaft aktivieren, lesen Clients aus Replikaten. Diese Eigenschaft gilt für den Typ EMBEDDED_PARTITION. Der Standardwert ist FALSE für den Typ EMBEDDED_PARTITION. Beim Typ EMBEDDED wird **ReplicaReadEnabled** immer auf TRUE gesetzt.

MaxUsedMemory

Gültige Werte: TRUE oder FALSEAktiviert das Entfernen von Cacheeinträgen, wenn ein Speicherengpass auftritt. Der Standardwert ist TRUE und sorgt dafür, dass Daten entfernt werden, wenn die Auslastung des JVM-Heap-Speichers den Schwellenwert von 70 % überschreitet. Sie können den Prozentsatz für den Schwellwert für die Auslastung des JVM-Heap-Speichers ändern, indem Sie die Eigenschaft "memoryThresholdPercentage" in der Datei objectGridServer.properties definieren und diese Datei in den Klassenpfad stellen. Weitere Informationen zu Evictor finden Sie in der Beschreibung der Evictor in der *Produktübersicht*. Weitere Informationen zur Servereigenschaftendatei finden Sie in der Beschreibung der ""im *Administratorhandbuch*.

MaxNumberOfReplicas

Gültige Werte: Größer-gleich 1Gibt die maximale Anzahl der für den Cache zu verwendenden Replikate an. Dieser Wert gilt nur für den Typ EMBEDDED. Der muss größer-gleich der Anzahl an Java Virtual Machines in einem System sein. Der Standardwert ist 47.

Die Eigenschaften "NumberOfPartitions", "ReplicaMode", "ReplicaReadEnabled" und "MaxNumberOfReplicas" sind Faktoren für die ObjectGrid-Implementierung. Die Eigenschaften "NumberOfPartitions", "ReplicaMode" und "ReplicaReadEnabled" gelten für den Typ EMBEDDED_PARTITION. ReplicaMode und dMaxNumberOfReplicas gelten beide für den Typ EMBEDDED.

Hinweise zu den Typen EMBEDDED und EMBEDDED_PARTITION

Die integrierten ObjectGrid-Typen verwenden die zuvor beschriebenen Konfigurationseigenschaften, um eine Gruppe von ObjectGrid-Containerservern und bei Bedarf einen Katalogservice zu konfigurieren und zu implementieren. Der Lebenszyklus der Container ist an die JPA-Anwendung gebunden und wird im Klassenpfad der Anwendung zusammengefasst. Wenn eine Anwendung gestartet wird, erkennt oder startet das Plug-in automatisch einen Katalogservice, startet einen Container und stellt die Verbindung zum Katalogservice her. Das Plug-in kommuniziert anschließend mit dem ObjectGrid-Container und seinen Peers, die in anderen Anwendungsserverprozessen ausgeführt werden, über die Clientverbindung.

Jeder JPA-Entität wird über den Klassennamen der Entität eine unabhängige BackingMap zugeordnet. Jede BackingMap hat die folgenden Attribute:

- readOnly="false"
- copyKey="false"

- `lockStrategy="NONE"`
- `copyMode="NO_COPY"`

Anmerkung: Wenn Sie den ObjectGridTyp-Wert `EMBEDDED` oder `EMBEDDED_PARTITION` in einer Java-SE-Umgebung verwenden, verwenden Sie am Ende des Programms die Methode "System.exit(0)", um den integrierten eXtreme-Scale-Server zu stoppen. Andernfalls scheint es so, als würde das Programm nicht reagieren.

ObjectGridType-Standardwerte

Der ObjectGridType-Wert gibt die Topologie an, in der der ObjectGrid-Cache implementiert wird. Der Standardtyp und der Typ, der die beste Leistung bietet, ist EMBEDDED. In den folgenden Abschnitten werden die Standardeigenschaften für jeden der ObjectGridType-Werte beschrieben.

Standardwerte für die ObjectGrid-JPA-Cachetopologie EMBEDDED

Wenn Sie den ObjectGrid-Typ `EMBEDDED` verwenden, werden die folgenden Standardwerte verwendet, wenn Sie keine Werte in der Konfiguration angeben:

- **ObjectGridName**: Name der Persistenzeinheit
- **ObjectGridType**: `EMBEDDED`
- **NumberOfPartitions**: `1` (kann nicht geändert werden, wenn der ObjectGrid-Typ `EMBEDDED` ist)
- **ReplicaMode**: `SYNC`
- **ReplicaReadEnabled**: `TRUE` (kann nicht geändert werden, wenn der ObjectGrid-Typ `EMBEDDED` ist)
- **MaxUsedMemory**: `TRUE`
- **MaxNumberOfReplicas**: `47` (muss kleiner-gleich der Anzahl Java Virtual Machines in einem verteilten System sein)

Sie müssen einen eindeutigen ObjectGridName-Wert angeben, um Namensunverträglichkeiten zu vermeiden. Der MaxNumberOfReplicas-Wert muss größer-gleich der Gesamtanzahl Java Virtual Machines im System sein.

ObjectGrid-Cachetopologie REMOTE

Der ObjectGrid-Typ REMOTE erfordert keine Eigenschaftseinstellungen, weil das ObjectGrid und die Implementierungsrichtlinie gesondert von der JPA-Anwendung definiert werden. Das JPA-Cache-Plug-in stellt über Fernzugriff eine Verbindung zu einem vorhandenen fernen ObjectGrid her.

Da alle Interaktionen mit dem ObjectGrid über Fernzugriff erfolgen, hat diese Topologie die geringste Leistung von allen ObjectGrid-Typen.

Hinweise zum Katalogservice und Konfiguration

Wenn Sie mit einer Topologie des Typs `EMBEDDED` oder `EMBEDDED_PARTITION` arbeiten, startet das JPA-Cache-Plug-in bei Bedarf automatisch einen einzigen Katalogservice in einem der Anwendungsserverprozesse. In einer Produktionsumgebung müssen Sie eine Katalogservicedomäne erstellen. Weitere Informationen zum Definieren eines Katalogservice finden Sie .in der Beschreibung des Katalogservice mit hoher Verfügbarkeit in der *Produktübersicht*.

Wenn Sie in einem Prozess von WebSphere Application Server arbeiten, stellt das JPA-Cache-Plug-in automatisch eine Verbindung zum Katalogservice bzw. zur Katalogservicedomäne her, der bzw. das für die Zelle von WebSphere Application Server definiert ist. Weitere Informationen zum Definieren einer Katalogservicedomäne finden Sie in den Informationen zum Starten des Katalogservice im *Administratorhandbuch*.

Wenn Sie Ihre Server nicht in einem Prozess von WebSphere Application Server ausführen, werden die Hosts und Ports der Katalogservicedomäne über eine Eigenschaftendatei mit dem Namen `objectGridServer.properties` angegeben. Diese Datei muss im Klassenpfad der Anwendung gespeichert werden und die definierte Eigenschaft **catalogServiceEndPoints** haben. Das Katalogservice-Grid wird unabhängig von den Anwendungsprozessen gestartet und muss vor den Anwendungsprozessen gestartet werden.

Das Format der Datei `objectGridServer.properties` ist wie folgt:

```
catalogServiceEndPoints=<Hostname1>:<Port1>,<Hostname2>:<Port2>
```

Cache-Plug-in für JPA

WebSphere eXtreme Scale enthält Cache-Plug-ins der Stufe 2 (L2) für die JPA-Provider OpenJPA und Hibernate.

Die Verwendung von eXtreme Scale als L2-Cacheprovider erhöht die Leistung beim Lesen und Abfragen von Daten und reduziert die Last der Datenbank. WebSphere eXtreme Scale bietet im Vergleich mit integrierten Cacheimplementierungen verschiedene Vorteile, weil der Cache automatisch in allen Prozessen repliziert wird. Wenn ein Client einen Wert zwischenspeichert, können alle anderen Clients den zwischengespeicherten Wert, der sich lokal im Speicher befindet, verwenden.

Mit den ObjectGrid-Cache-Plug-ins für OpenJPA und Hibernate können Sie drei Topologietypen erstellen: integriert, integriert-partitioniert und fern.

Integrierte Topologie

Eine integrierte Topologie erstellt einen eXtreme-Scale-Server in dem Prozessbereich jeder Anwendung. OpenJPA und Hibernate lesen die Speicherkopie des Caches direkt und schreiben in alle anderen Kopien. Sie können die Schreibleistung durch den Einsatz asynchroner Replikation verbessern. Diese Standardtopologie liefert die beste Leistung, wenn die zwischengespeicherte Datenmenge in einen einzigen Prozess passt.

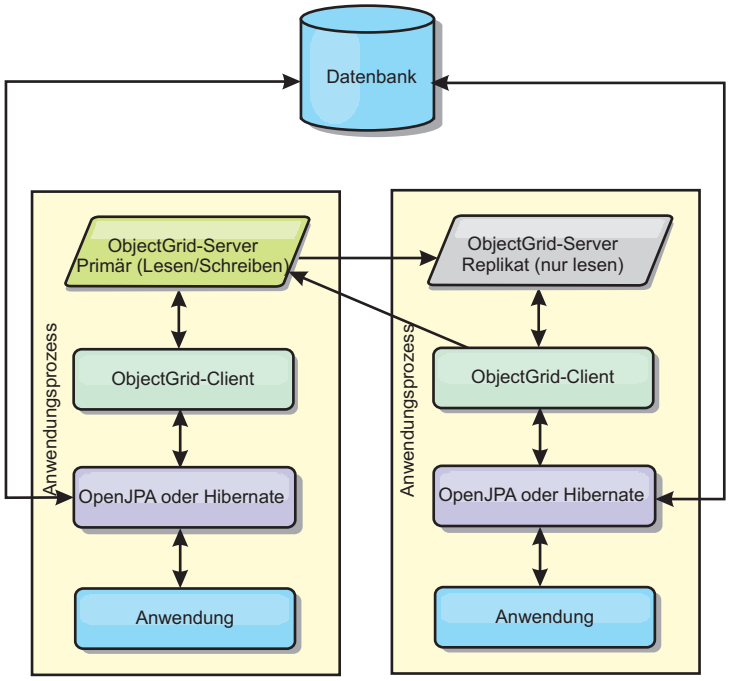


Abbildung 11. Integrierte JPA-Topologie

Vorteile:

- Alle Leseoperationen im Cache sind sehr schnelle lokale Zugriffe.
- Die Konfiguration ist einfach.

Einschränkungen:

- Das Datenvolumen ist auf die Größe des Prozesses beschränkt.
- Alle Cacheaktualisierungen werden an einen einzigen Prozess gesendet.

Integrierte, partitionierte Topologie

Wenn die zwischengespeicherten Daten nicht in einen einzigen Prozess passen, verwendet die integrierte, partitionierte Topologie ObjectGrid-Partitionen, um die Daten auf mehrere Prozesse zu verteilen. Die Leistung ist nicht so hoch wie bei der integrierten Topologie, weil die meisten Leseoperationen im Cache über Fernzugriff durchgeführt werden. Sie können diese Option trotzdem verwenden, wenn die Latenzzeit der Datenbank hoch ist.

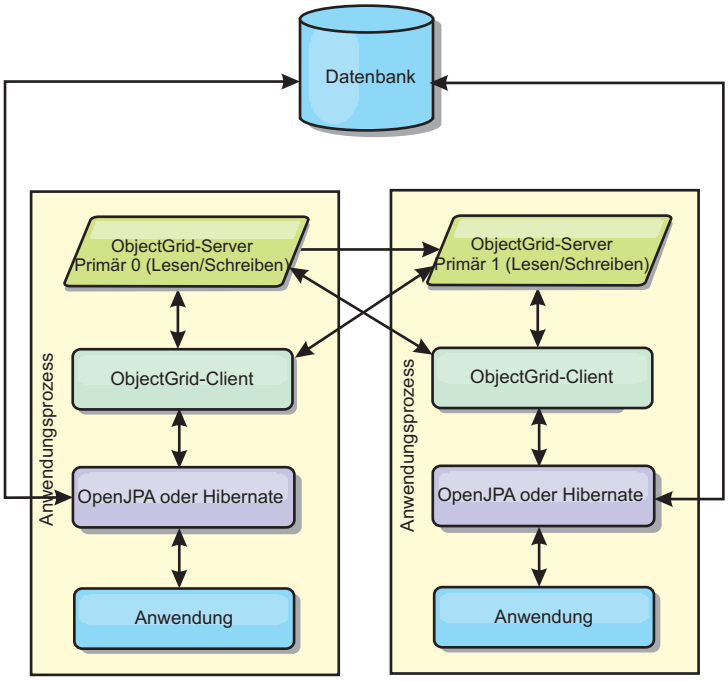


Abbildung 12. Integrierte, partitionierte JPA-Topologie

Vorteile:

- Es können große Datenvolumen gespeichert werden.
- Die Konfiguration ist einfach.
- Cacheaktualisierungen werden auf mehrere Prozesse verteilt.

Einschränkungen:

- Die meisten Lese- und Aktualisierungsoperationen im Cache werden über Fernzugriff durchgeführt.

Um beispielsweise 10 GB Daten mit maximal 1 GB pro JVM zu speichern, sind zehn Java Virtual Machines erforderlich. Die Anzahl der Partitionen muss daher auf mindestens 10 gesetzt werden. Im Idealfall wird die Anzahl der Partitionen auf eine Primzahl gesetzt, so dass in jedem Shard eine angemessene Speichermenge zugeteilt wird. Gewöhnlich entspricht der Wert der Einstellung "numberOfPartitions" der Anzahl der Java Virtual Machines. Bei dieser Einstellung enthält jede JVM eine Partition. Wenn Sie die Replikation aktivieren, müssen Sie die Anzahl der Java Virtual Machines im System erhöhen. Andernfalls wird in jeder JVM zusätzlich eine Replikatpartition gespeichert, die genauso viel Speicher belegt wie eine primäre Partition.

Lesen Sie die Informationen zur Berechnung der Speicherkapazität und der Partitionsanzahl im *Administratorhandbuch*, um die Leistung der von Ihnen ausgewählten Konfiguration zu maximieren.

In einem System mit vier Java Virtual Machines und einem numberOfPartitions-Wert von 4 beispielsweise enthält jede JVM eine primäre Partition. Bei einer Leseoperation besteht eine Chance von 25 %, dass die Daten aus einer lokal verfügbaren Partition abgerufen werden, was im Vergleich mit dem Abruf der Daten aus einer fernen JVM wesentlich schneller ist. Wenn eine Leseoperation, z. B. eine Abfrage, eine Sammlung von Daten abrufen muss, die gleichmäßig auf vier Partitionen verteilt sind, sind 75 % der Aufrufe ferne und 25 % der Aufrufe lokale Aufru-

fe. Wenn die Einstellung "ReplicaMode" auf SYNC oder ASYNC und die Einstellung "ReplicaReadEnabled" auf true gesetzt wird, werden vier Relikatpartitionen erstellt und auf vier Java Virtual Machines verteilt. Jede JVM enthält eine primäre Partition und eine Replikatpartition. Die Chance, dass die Leseoperation lokal ausgeführt wird, erhöht sich auf 50 %. Die Leseoperation, die eine Sammlung von Daten abgerufen muss, die gleichmäßig auf vier Partitionen verteilt sind, hat 50 % ferne Aufrufe und 50% lokale Aufrufe. Lokale Aufrufe sind wesentlich schneller als ferne Aufrufe. Mit jedem fernen Aufruf nimmt die Leistung ab.

Ferne Topologie

In einer fernen Topologie werden alle zwischengespeicherten Daten in einem oder mehreren gesonderten Prozessen gespeichert, was die Speicherbelegung der Anwendungsprozesse verringert. Sie können Ihre Daten auf unterschiedliche Prozesse verteilen, indem Sie ein partitioniertes, repliziertes eXtreme-Scale-Grid implementieren. Im Gegensatz zu den integrierten und integrierten partitionierten Konfigurationen, die in den vorherigen Abschnitten beschrieben wurden, müssen Sie ein fernes Grid unabhängig von der Anwendung und vom JPA-Provider verwalten. Weitere Einzelheiten zur Verwaltung einer Implementierung eines eXtreme-Scale-Grids finden Sie in den Informationen zur Überwachung von Implementierungsumgebungen.

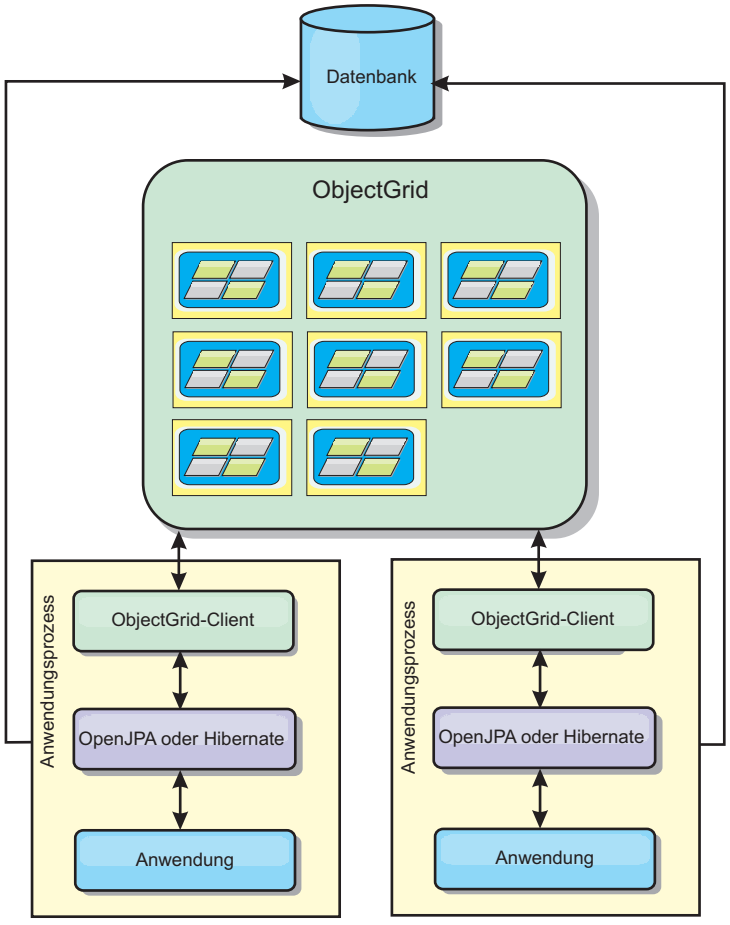


Abbildung 13. Ferne JPA-Topologie

Vorteile:

- Es können große Datenvolumen gespeichert werden.

- Der Anwendungsprozess ist frei von zwischengespeicherten Daten.
- Cacheaktualisierungen werden auf mehrere Prozesse verteilt.
- Sehr flexible Konfigurationsoptionen.

Einschränkungen:

- Alle Lese- und Aktualisierungsoperationen im Cache werden über Fernzugriff durchgeführt.

Konfiguration eines Hibernate-Cache-Plug-ins

Ein eXtreme-Scale-Cache kann für Hibernate aktiviert werden, indem Eigenschaften in der Konfigurationsdatei definiert werden.

7.1+ Für die Integration mit WebSphere Application Server wird das Hibernate-Cache-Plug-in in die Datei `oghibernate-cache.jar` gepackt und in `WAS_HOME/optionalLibraries/ObjectGrid` installiert. Wenn Sie das Hibernate-Cache-Plug-in verwenden möchten, müssen Sie die Datei `oghibernate-cache.jar` in die Hibernate-Bibliothek einfügen. Wenn Sie die Hibernate-Bibliothek beispielsweise in Ihre Anwendung einfügen, müssen Sie auch die Datei `oghibernate-cache.jar` einfügen. Wenn Sie eine gemeinsam genutzte Bibliothek definieren, um die Hibernate-Bibliothek einzufügen, müssen Sie die Datei `oghibernate-cache.jar` im Verzeichnis der gemeinsam genutzten Bibliothek ablegen.

7.1+ eXtreme Scale Version 7.1 installiert die Datei "cglib.jar" nicht in der Umgebung von WebSphere Application Server. Wenn Sie vorhandene Anwendungen oder gemeinsam genutzte Bibliotheken haben wie Hibernate, die von der Datei `cglib.jar` abhängig sind, suchen Sie die Datei `cglib.jar`, und fügen Sie sie dem Klassenpfad hinzu. Enthält Ihre Anwendung beispielsweise alle JAR-Dateien für die Hibernate-Bibliothek, aber nicht die für Hibernate verfügbare Datei `cglib.jar`, müssen Sie die Datei `cglib.jar` von Hibernate in Ihre Anwendung einfügen.

Einstellungen

Im Folgenden sehen Sie die Syntax für die Definition der Eigenschaft in der Datei `persistence.xml`:

persistence.xml

```
<property name="hibernate.cache.provider_class"
         value="com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider" />
<property name="hibernate.cache.use_query_cache" value="true"/>
<property name="objectgrid.configuration" value="<Eigenschaft>=<Wert>,..." />
<property name="objectgrid.hibernate.regionNames" value="<Regionsname>,.." />
```

Im Folgenden sehen Sie die Syntax für die Definition der Eigenschaft in der Datei `hibernate.cfg.xml`:

hibernate.cfg.xml

```
<property name="cache.provider_class">com.ibm.websphere.objectgrid.
          hibernate.cache.ObjectGridHibernateCacheProvider</property>
<property name="cache.use_query_cache">true</property>
<property name="objectgrid.configuration"><Eigenschaft>=<Wert>,...</property>
<property name="objectgrid.hibernate.regionNames"><Regionsname>,...</property>
```

Die Eigenschaft für "provider_class" ist "com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider". Zum Aktivieren des Abfragecaches setzen Sie den Wert der Eigenschaft "use_query_cache" auf `true`. Verwenden Sie die Eigenschaft "objectgrid.configuration", um die Konfigurationseigenschaften für den eXtreme-Scale-Cache festzulegen.

Sie müssen einen eindeutigen Wert für die Eigenschaft "ObjectGridName" angeben, um potenzielle Namenskonflikte zu vermeiden. Die anderen Konfigurationseigenschaften für den eXtreme-Scale-Cache sind optional.

Die Eigenschaft "objectgrid.hibernate.regionNames" ist optional und muss angegeben werden, wenn nach der Initialisierung des eXtreme-Scale-Caches regionsNames-Werte definiert werden. Angenommen, eine Entitätsklasse ist einem regionName-Wert zugeordnet, und die Entitätsklasse ist weder in der Datei `persistence.xml` noch in der Hibernate-Zuordnungsdatei enthalten. Weiter angenommen, die Entitätsklasse hat eine Annotation "Entity". In diesem Fall wird der regionName-Wert für diese Entitätsklasse beim Laden der Klassen aufgelöst, wenn der eXtreme-Scale-Cache initialisiert wird. Ein weiteres Beispiel ist die Methode "Query.setCacheRegion(String regionName)", die nach der Initialisierung des Caches von eXtreme Scale ausgeführt wird. In diesen Situationen müssen Sie alle möglichen dynamisch bestimmten Regionsnamen (regionNames) in die Eigenschaft "objectgrid.hibernate.regionNames" einfügen, damit der eXtreme-Scale-Cache BackingMaps für alle Regionsnamen vorbereiten kann.

Im Folgenden sehen Sie Beispiele für die Dateien `persistence.xml` und `hibernate.cfg.xml`:

persistence.xml

```
<persistence-unit name="testPU2">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <class>com.ibm.websphere.objectgrid.jpa.test.Department</class>
    <properties>
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.connection.url" value="jdbc:derby:DB_testPU2;create=true" />
      <property name="hibernate.connection.driver_class" value="org.apache.derby.jdbc.EmbeddedDriver" />

      <property name="hibernate.cache.provider_class" value="com.ibm.websphere.objectgrid.
               hibernate.cache.ObjectGridHibernateCacheProvider" />
      <property name="hibernate.cache.use_query_cache" value="true"/>
      <property name="objectgrid.configuration" value="ObjectGridName=myOGName,ObjectGridType=
               EMBEDDED,MaxNumberOfReplicas=4" writeBehind=true, writeBehindInterval=5000,
               writeBehindPoolSize=10, writeBehindMaxBatchSize=1000" />
      <property name="objectgrid.hibernate.regionNames" value="queryRegion1, queryRegion2" />
    </properties>
</persistence-unit>
```

7.1+ Version 7.1 führt zusätzlich zu den Standardkonfigurationsoptionen für JPA-Cache-Plug-ins weitere spezielle Konfigurationsoptionen für die Write-behind-Funktion für das Hibernate-Cache-Plug-in ein.

writeBehind

Gültige Werte: TRUE und FALSE

Standardwert: FALSE

Wenn writeBehind aktiviert ist, werden Aktualisierungen vorübergehend in einem JVM-spezifischen Datenspeicher gespeichert, bis die writeBehindInterval- bzw. writeBehindMaxBatchSize-Bedingung erfüllt ist.

Achtung: Wenn writeBehind nicht aktiviert ist, werden weitere Write-behind-Konfigurationseinstellungen ignoriert.

writeBehindInterval

Gültige Werte: Größer-gleich 1

Standardwert 5000 (5 Sekunden)

Gibt das Zeitintervall (in Millisekunden) an, in dem Aktualisierungen mit einer Flush-Operation in den Cache geschrieben werden.

writeBehindPoolSize

Gültige Werte: Größer-gleich 1

Standardwert: 5

Gibt die maximale Größe des Thread-Pools an, der für das Schreiben von Aktualisierungen in den Cache verwendet wird.

writeBehindMaxBatchSize

Gültige Werte: Größer-gleich 1

Standardwert: 1000

Gibt die maximale Stapelgröße por Regionscache für das Schreiben von Aktualisierungen in den Cache an.

Der vorherige Beispielcode zeigt die folgende Konfiguration für die Write-behind-Funktion an:

```
writeBehind=true, writeBehindInterval=5000, writeBehindPoolSize=10, writeBehindMaxBatchSize=1000
```

Für diese Angaben gilt Folgendes:

- writeBehind=TRUE aktiviert die Write-behind-Funktion.
- writeBehindInterval=5000 bedeutet, dass Aktualisierungen in einem Intervall von ca. 5 Sekunden mit einer Flush-Operation in den Cache geschrieben werden.
- writeBehindPoolSize=10 bedeutet, dass maximal 10 Threads für das Schreiben von Aktualisierungen in den Cache eingesetzt werden.
- writeBehindMaxBatchSize=1000 bedeutet, dass die Aktualisierungen mit einer Flush-Operation in den Cache geschrieben werden, wenn die Anzahl der im Write-behind-Speicher eines Regionscaches gespeicherten Einträge den Wert 1000 überschreitet, und zwar auch dann, wenn die angegebene writeBehindInterval-Bedingung nicht erfüllt ist. Anders ausgedrückt, die Aktualisierungen werden entweder alle fünf Sekunden oder bei mehr als 1000 Einträgen im Write-behind-Speicher jedes Regionscaches mit einer Flush-Operation in den Cache geschrieben. Wenn die writeBehindMaxBatchSize-Bedingung erfüllt ist, schreibt nur der Regionscache, der diese Bedingung erfüllt, seine Aktualisierungen im Write-behind-Speicher mit einer Flush-Operation in den Cache. Ein Regionscache entspricht gewöhnlich einer Entität oder einer Abfrage.

Wichtig:

Verwenden Sie die Write-behind-Funktionskonfiguration mit Vorsicht. Sie führt zu längeren Latenzzeiten bei der Datensynchronisation in allen JVMs und zu einem höheren Risiko von Aktualisierungsverlusten. In einem System, das die Write-behind-Konfiguration mit vier oder mehr JVMs verwendet, wird die in einer JVM durchgeführte Aktualisierung mit einer Verzögerung von ca. 15 Sekunden anderen JVMs bereitgestellt. Wenn zwei JVMs denselben Eintrag akualisieren, verliert die JVM, die die Aktualisierung zuerst mit einer Flush-Operation in den Cache schreibt, ihre Aktualisierung.

hibernate.cfg.xml

```
<hibernate-configuration>
    <session-factory>
        <!-- Database connection settings -->
        <property name="connection.driver_class">org.apache.derby.jdbc.EmbeddedDriver</property>
        <property name="connection.url">jdbc:derby:DB_testPU2;create=true</property>
        <!-- ObjectGrid cache setting-->
        <property name="cache.provider_class">com.ibm.websphere.objectgrid.hibernate.cache.
                ObjectGridHibernateCacheProvider</property>
        <property name="cache.use_query_cache">true</property>
        <property name="objectgrid.configuration">ObjectGridName=myOGName,
                 ObjectGridType=EMBEDDED,MaxNumberOfReplicas=4 </property>
        <property name="objectgrid.hibernate.regionNames">queryRegion1, queryRegion2</property>
```

```
        <mapping resource="com/ibm/websphere/objectgrid/jpa/test/Employee.hbm.xml"/>

    </session-factory>
</hibernate-configuration>
```

Daten vorab in den ObjectGrid-Cache laden

Sie können die Methode "preload" der Klasse "ObjectGridHibernateCacheProvider" verwenden, um Daten für eine Entitätsklasse vorab in den ObjectGrid-Cache zu laden.

Beispiel 1

Mit EntityManagerFactory

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("testPU");
ObjectGridHibernateCacheProvider.preload("objectGridName", emf, TargetEntity.class, 100, 100);
```

Beispiel 2

Mit SessionFactory

```
org.hibernate.cfg.Configuration cfg = new Configuration();
// Methoden addResource, addClass, und setProperty von Configuration verwenden,
// um erforderliche Konfiguration zum Erstellen von SessionFactor vorzubereiten
SessionFactory sessionFactory= cfg.buildSessionFactory();
ObjectGridHibernateCacheProvider.preload("objectGridName", sessionFactory, TargetEntity.class, 100, 100);
```

Anmerkung:

1. In einem verteilten System kann dieser Preload-Mechanismus nur über eine einzige Java Virtual Machine aufgerufen werden. Der Preload-Mechanismus kann nicht gleichzeitig über mehrere JVMs ausgeführt werden.
2. Vor der Ausführung des Preload-Mechanismus müssen Sie den eXtreme-Scale-Cache initialisieren, indem Sie einen EntityManager über die EntityManagerFactory erstellen, damit alle entsprechenden BackingMaps erstellt werden. Andernfalls zwingt der Preload-Mechanismus die Initialisierung des Caches mit einer einzigen Standard-BackingMap für die Unterstützung aller Entitäten. Das bedeutet, dass eine einzige BackingMap von allen Entitäten gemeinsam genutzt wird.

Hibernate-Cachekonfiguration mit XML anpassen

Für die meisten Szenarien reicht die Definition von Cacheeigenschaften aus. Wenn Sie das vom Cache verwendete ObjectGrid weiter anpassen möchten, können Sie Hibernate-ObjectGrid-XML-Konfigurationsdateien im Verzeichnis `META-INF` bereitstellen, wie z. B. die Datei `persistence.xml`. Während der Initialisierung versucht der Cache, diese XML-Dateien zu finden und sie dann zu verarbeiten.

Es gibt drei Typen von Hibernate-ObjectGrid-XML-Konfigurationsdateien: `hibernate-objectGrid.xml` (ObjectGrid-Konfiguration), `hibernate-objectGridDeployment.xml` (Implementierungsrichtlinie) und `hibernate-objectGrid-client-override.xml` (Konfiguration der ObjectGrid-Clientkorrekturwerte). Je nach konfigurierter eXtreme-Scale-Topologie können Sie jede dieser drei XML-Dateien verwenden, um diese Topologie anzupassen.

Für die Typen EMBEDDED und EMBEDDED_PARTITION können Sie jede der drei XML-Dateien für die Anpassung des ObjectGrids, der Implementierungsrichtlinie oder der Konfiguration der ObjectGrid-Clientkorrekturwerte verwenden.

Bei ObjectGrids des Typs REMOTE erstellt der Cache kein dynamisches ObjectGrid. Vielmehr ruft der Cache ein clientseitiges ObjectGrid vom Katalogservice ab.

Zum Anpassen der Konfiguration für die ObjectGrid-Clientkorrekturwerte können Sie nur eine Datei hibernate-objectGrid-client-override.xml verwenden.

1. **ObjectGrid-Konfiguration**: Verwenden Sie die Datei META-INF/hibernate-objectGrid.xml. Diese Datei wird verwendet, um die ObjectGrid-Konfiguration für die Typen EMBEDDED und EMBEDDED_PARTITION anzupassen. Für den Typ REMOTE wird diese Datei ignoriert. Standardmäßig hat jede Entitätsklasse einen zugeordneten Regionsnamen (standardmäßig den Namen der Entitätsklasse), der einer BackingMap-Konfiguration zugeordnet ist, die nach dem Regionsnamen in der ObjectGrid-Konfiguration benannt ist. Die Entitätsklasse "com.mycompany.Employee" hat beispielsweise standardmäßig einen zugeordneten Regionsnamen, der der BackingMap "com.mycompany.Employee" zugeordnet ist. Die BackingMap-Standardkonfiguration hat die Einstellungen readOnly="false", copyKey="false", lockStrategy="NONE" und copyMode="NO_COPY". Sie können einige BackingMaps mit einer ausgewählten Konfiguration anpassen. Das reservierte Schlüsselwort "ALL_ENTITY_MAPS" kann verwendet werden, um alle Maps mit Ausnahme der angepassten Maps aus der Datei hibernate-objectGrid.xml darzustellen. BackingMaps, die nicht in der Datei hibernate-objectGrid.xml aufgelistet sind, verwenden die Standardkonfiguration.
2. **ObjectGridDeployment-Konfiguration**: Verwenden Sie die Datei META-INF/hibernate-objectGridDeployment.xml. Diese Datei wird verwendet, um die Implementierungsrichtlinie anzupassen. Wenn Sie bei der Anpassung der Implementierungsrichtlinie die Datei hibernate-objectGridDeployment.xml bereitstellen, wird die Standardimplementierungsrichtlinie verworfen. Alle Attributwerte für die Implementierungsrichtlinie werden der bereitgestellten Datei hibernate-objectGridDeployment.xml entnommen.
3. **Konfiguration der ObjectGrid-Clientkorrekturwerte**: Verwenden Sie die Datei META-INF/hibernate-objectGrid-client-override.xml. Diese Datei wird verwendet, um ein clientseitiges ObjectGrid anzupassen. Standardmäßig wendet der ObjectGrid-Cache eine Standardkonfiguration für Clientkorrekturwerte an, die den nahen Cache inaktiviert. Wenn eine Anwendung einen nahen Cache erfordert, können Sie diese Datei bereitstellen und numberOfBuckets="xxx" angeben. Der Standardclientkorrekturwert inaktiviert den nahen Cache mit numberOfBuckets="0". Der nahe Cache kann aktiv sein, wenn das Attribut "numberOfBuckets" über die Datei hibernate-objectGrid-client-override.xml auf einen Wert größer als 0 zurückgesetzt wird. Die Funktionsweise der Datei hibernate-objectGrid-client-override.xml gleicht der der Datei hibernate-objectGrid.xml: Sie überscheibt oder erweitert die Standardkonfiguration für ObjectGrid-Clientkorrekturwerte.

Beispiele für Hibernate-ObjectGrid-XML-Dateien

Hibernate-ObjectGrid-XML-Dateien müssen auf der Basis einer Persistenzeinheit erstellt werden.

Im Folgenden sehen Sie eine Beispieldatei persistence.xml, die die Konfiguration einer Persistenzeinheit darstellt:

persistence.xml

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  version="1.0">
  <persistence-unit name="AnnuityGrid">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>

    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.FixedAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.EquityAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider</class>
```

```
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Person</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityHolder</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Address</class>

    <exclude-unlisted-classes>true</exclude-unlisted-classes>

    <properties>
      <property name="hibernate.show_sql" value="false" />
      <property name="hibernate.connection.url" value="jdbc:db2:Annuity" />
      <property name="hibernate.connection.driver_class" value="com.ibm.db2.jcc.DB2Driver" />
      <property name="hibernate.default_schema" value="EJB30" />

      <!-- Cache -->
      <property name="hibernate.cache.provider_class"
        value="com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider" />
      <property name="hibernate.cache.use_query_cache" value="true" />
      <property name="objectgrid.configuration" value="ObjectGridType=EMBEDDED,
               ObjectGridName=Annuity, MaxNumberOfReplicas=4" />
    </properties>
  </persistence-unit>

</persistence>
```

Im Folgenden sehen Sie die Datei `hibernate-objectGrid.xml`, die der Datei `persistence.xml` entspricht:

hibernate-objectGrid.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="Annuity">
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <backingMap name="defaultCacheMap" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="defaultCacheMap" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
      <backingMap name="org.hibernate.cache.UpdateTimestampsCache" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="org.hibernate.cache.UpdateTimestampsCache" />
      <backingMap name="org.hibernate.cache.StandardQueryCache" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="org.hibernate.cache.StandardQueryCache" />
    </objectGrid>
  </objectGrids>
  <backingMapPluginCollections>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="defaultCacheMap">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
```

```
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="org.hibernate.cache.UpdateTimestampsCache">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>


    </backingMapPluginCollection>
    <backingMapPluginCollection id="org.hibernate.cache.StandardQueryCache">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Anmerkung: Die Maps "org.hibernate.cache.UpdateTimestampsCache", "org.hibernate.cache.StandardQueryCache" und "defaultCacheMap" sind erforderlich.

Es folgt die Datei `hibernate-objectGridDeployment.xml`, die der Datei `persistence.xml` entspricht:

hibernate-objectGridDeployment.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="Annuity">
    <mapSet name="MAPSET_Annuity" numberOfPartitions="1" numInitialContainers="1" minSyncReplicas="0"
          maxSyncReplicas="4" maxAsyncReplicas="0" replicaReadEnabled="true">
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <map ref="defaultCacheMap" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
      <map ref="org.hibernate.cache.UpdateTimestampsCache" />
      <map ref="org.hibernate.cache.StandardQueryCache" />
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

Anmerkung: Die Maps "org.hibernate.cache.UpdateTimestampsCache", "org.hibernate.cache.StandardQueryCache" und "defaultCacheMap" sind erforderlich.

Externes System für einen Cache mit dem ObjectGrid-Typ REMOTE

Sie müssen ein externes eXtreme-Scale-System einrichten, wenn Sie einen Cache mit dem ObjectGrid-Typ REMOTE konfigurieren möchten. Sie benötigen ObjectGrid- und ObjectGridDeployment-XML-Konfigurationsdateien, die auf der Datei `persistence.xml` basieren, um ein externes System einrichten zu können. Die Hibernate-ObjectGrid- und ObjectGridDeployment-XML-Konfigurationsdateien, die im Abschnitt mit den Beispielen für Hibernate-ObjectGrid-XML-Dateien beschrieben werden, können auch zum Einrichten eines externen eXtreme-Scale-Systems verwendet werden.

Ein externes ObjectGrid-System hat sowohl Katalogservice- als auch ObjectGrid-Serverprozesse. Sie müssen einen Katalogservice starten, bevor Sie Containerserver starten. Weitere Informationen finden Sie in den Einzelheiten zum Starten eigenständiger Server von eXtreme Scale und Containerprozessen im *Administratorhandbuch*.

Fehlerbehebung

1. **`CacheException: Failed to get ObjectGrid server`**

   Bei einem ObjectGrid des Typs EMBEDDED oder EMBEDDED_PARTITION versucht der Cache, eine Serverinstanz von der Laufzeitumgebung von eXtreme Scale abzurufen. In einer Java-SE-Umgebung wird ein Server von eXtreme Scale

mit integriertem Katalogservice gestartet. Der integrierte Katalogservice versucht, an Port 2809 empfangsbereit zu sein. Wenn dieser Port von einem anderen Prozess verwendet wird, tritt dieser Fehler auf. Wenn externe Katalogserviceendpunkte angegeben werden, z. B. in der Datei `objectGridServer.properties`, tritt dieser Fehler auf, wenn der Hostname oder Port falsch angegeben sind.

2. **CacheException: Failed to get REMOTE ObjectGrid for configured REMOTE ObjectGrid. objectGridName = [ObjectGridName], PU name = [persistenceUnitName]**

   Dieser Fehler tritt auf, wenn der Cache kein ObjectGrid von den bereitgestellten Katalogserviceendpunkten abrufen kann. Gewöhnlich ist dieser Fehler auf einen falsch angegebenen Hostnamen oder Port zurückzuführen.

3. **CacheException: Cannot have two PUs [persistenceUnitName_1, persistenceUnitName_2] configured with same ObjectGridName [ObjectGridName] of EMBEDDED ObjectGridType**

   Diese Ausnahme tritt ein, wenn Sie eine Konfiguration mit vielen Persistenzeinheiten haben und die Caches dieser Einheiten mit demselben ObjectGrid-Namen und dem ObjectGrid-Typ EMBEDDED konfiguriert sind. Diese Persistenzeinheitenkonfigurationen können in derselben oder in unterschiedlichen Dateien `persistence.xml` enthalten sein. Sie müssen sicherstellen, dass der ObjectGrid-Name für jede Persistenzeinheit eindeutig ist, wenn der ObjectGrid-Typ EMBEDDED verwendet wird.

4. **CacheException: REMOTE ObjectGrid [ObjectGridName] does not include required BackingMaps [mapName_1, mapName_2,...]**

   Wenn der ObjectGrid-Typ REMOTE verwendet wird und das abgerufene clientseitige ObjectGrid keine vollständigen Entitäts-BackingMaps für die Unterstützung des Caches der Persistenzeinheit hat, wird diese Ausnahme ausgelöst. Beispiel: Es sind fünf Entitätsklassen in der Konfiguration der Persistenzeinheit aufgelistet, aber das abgerufene ObjectGrid hat nur zwei BackingMaps. Diese Ausnahme wird auch dann ausgelöst, wenn das abgerufene ObjectGrid zehn BackingMaps enthält, aber eine der fünf erforderlichen Entitäts-BackingMaps nicht unter den zehn vorhandenen gefunden wird.

Konfiguration des OpenJPA-Cache-Plug-ins

WebSphere eXtreme Scale stellt DataCache- und QueryCache-Implementierungen für OpenJPA bereit. OpenJPA-ObjectGrid-Cache und OpenJPA-ObjectGrid-Cache sind allgemeine Kurzbezeichnungen für die DataCache- und QueryCache-Implementierungen.

Einstellungen

Der eXtreme-Scale-Cache wird für OpenJPA über die Konfigurationseigenschaften "openjpa.DataCache" und "openjpa.QueryCache" in der Datei `persistence.xml` aktiviert und inaktiviert. Die Syntax für die Einstellung der Eigenschaften ist wie folgt:

```
<property name="openjpa.DataCache"
          value="<object_grid_datacache_class(<Eigenschaft>=<Wert>,...)"/>
<property name="openjpa.QueryCache"
          value="<object_grid_querycache_class(<Eigenschaft>=<Wert>,...)"/>
```

DataCache und QueryCache können die Eigenschaften des eXtreme-Scale-Caches verwenden, um den von der Persistenzeinheit verwendeten Cache zu konfigurieren.

Zusätzlich zur Einstellung für den eXtreme-Scale-Cache muss die Eigenschaft "openjpa.RemoteCommitProvider" auf `sjvm` gesetzt werden:

```
<property name="openjpa.RemoteCommitProvider" value="sjvm"/>
```

Das mit der Annotation "@DataCache" für jede Entitätsklasse angegebene Zeitlimit wird an die BackingMap weitergegeben, in der jede Entität zwischengespeichert wird. Der mit der Annotation "@DataCache" angegebene Namenswert wird jedoch vom eXtreme-Scale-Cache ignoriert. Der vollständig qualifizierte Entitätsklassenname ist der Name der Cache-Map.

Die Methoden "pin" und "unpin" von OpenJPA-StoreCache und OpenJPA-QueryCache werden nicht unterstützt und haben keine Funktion.

Sie können die Eigenschaft "ObjectGridName", die Eigenschaft "ObjectGridType" und andere einfache auf die Implementierungsrichtlinie bezogene Eigenschaften in der Eigenschaftenliste der ObjectGrid-Cacheklasse angeben, um die Cachekonfiguration anzupassen. Es folgt ein Beispiel:

```
<property name="openjpa.DataCache"
          value="com.ibm.websphere.objectgrid.openjpa.ObjectGridDataCache(
          ObjectGridName=BasicTestObjectGrid,ObjectGridType=EMBEDDED,
          maxNumberOfReplicas=4)"/>
<property name="openjpa.QueryCache"
          value="com.ibm.websphere.objectgrid.openjpa.ObjectGridQueryCache()"/>
<property name="openjpa.RemoteCommitProvider" value="sjvm"/>
```

DataCache- und QueryCache-Konfigurationen sind voneinander unabhängig. Sie können beide Konfigurationen aktivieren. Wenn Sie jedoch beide Konfigurationen aktivieren, verwendet QueryCache dieselbe Konfiguration wie DataCache, und die Konfiguration von QueryCache wird verworfen.

OpenJPA-Cachekonfiguration mit XML anpassen

Für die meisten Szenarien ist die Definition der Eigenschaften des eXtreme-Scale-Caches ausreichend. Um das vom Cache verwendete ObjectGrid weiter anzupassen, können Sie ObjectGrid-XML-Konfigurationsdateien für OpenJPA wie die Datei `persistence.xml` in Ihrem Verzeichnis META-INF bereitstellen. Während der Cacheinitialisierung versucht der ObjectGrid-Cache, diese XML-Datei zu finden und dann zu verarbeiten.

Es gibt drei Typen von OpenJPA-ObjectGrid-XML-Konfigurationsdateien: `openjpa-objectGrid.xml` (ObjectGrid-Konfiguration), `openjpa-objectGridDeployment.xml` (Implementierungsrichtlinie) und `openjpa-objectGrid-client-override.xml` (Konfiguration der ObjectGrid-Clientkorrekturwerte). Je nach konfiguriertem ObjectGrid-Typ können Sie jede dieser Datei XML-Dateien verwenden, um das ObjectGrid anzupassen.

Für die Typen EMBEDDED und EMBEDDED_PARTITION können Sie jede der drei XML-Dateien für die Anpassung des ObjectGrids, der Implementierungsrichtlinie oder der Konfiguration der ObjectGrid-Clientkorrekturwerte verwenden.

Bei ObjectGrids des Typs REMOTE erstellt der ObjectGrid-Cache kein dynamisches ObjectGrid. Vielmehr ruft der Cache nur ein clientseitiges ObjectGrid vom Katalogservice ab. Zum Anpassen der Konfiguration für die ObjectGrid-Clientkorrekturwerte können Sie nur eine Datei `openjpa-objectGrid-client-override.xml` verwenden.

1. **ObjectGrid-Konfiguration**: Verwenden Sie die Datei `META-INF/openjpa-objectGrid.xml`. Diese Datei wird verwendet, um die ObjectGrid-Konfiguration für die Typen EMBEDDED und EMBEDDED_PARTITION anzupassen. Für den Typ REMOTE wird diese Datei ignoriert. Standardmäßig wird jede Entitätsklas-

se einer eigenen BackingMap-Konfiguration zugeordnet, die denselben Namen wie die Entitätsklasse in der ObjectGrid-Konfiguration hat. Die Entitätsklasse "com.mycompany.Employee" wird beispielsweise der BackingMap "com.mycompany.Employee" zugeordnet. Die BackingMap-Standardkonfiguration hat die Einstellungen `readOnly="false"`, `copyKey="false"`, `lockStrategy="NONE"` und `copyMode="NO_COPY"`. Sie können einige BackingMaps mit der ausgewählten Konfiguration anpassen. Sie können das reservierte Schlüsselwort `ALL_ENTITY_MAPS` verwenden, um alle Maps darzustellen mit Ausnahme anderer angepasster Maps, die in der Datei `openjpa-objectGrid.xml` aufgelistet sind. BackingMaps, die nicht in der Datei `openjpa-objectGrid.xml` aufgelistet sind, verwenden die Standardkonfiguration. Wenn für angepasste BackingMaps kein Attribut "BackingMaps" oder keine Eigenschaften definiert sind, diese Attribute aber in der Standardkonfiguration angegeben sind, werden die Attributwerte aus der Standardkonfiguration angewendet. Ist eine Entitätsklasse beispielsweise mit `timeToLive=30` annotiert, enthält die BackingMap-Standardkonfiguration für diese Entität ebenfalls `timeToLive=30`. Wenn die angepasste Datei `openjpa-objectGrid.xml` diese BackingMap ebenfalls enthält, aber den timeToLive-Wert nicht definiert, wird für die angepasste BackingMap standardmäßig der Wert `timeToLive=30` verwendet. Die Datei `openjpa-objectGrid.xml` soll die Standardkonfiguration überschreiben oder erweitern.

2. **ObjectGridDeployment-Konfiguration**: Verwenden Sie die Datei `META-INF/openjpa-objectGridDeployment.xml`. Diese Datei wird verwendet, um die Implementierungsrichtlinie anzupassen. Wenn Sie bei der Anpassung der Implementierungsrichtlinie die Datei `openjpa-objectGridDeployment.xml` bereitstellen, wird die Standardimplementierungsrichtlinie verworfen. Alle Attributwerte für die Implementierungsrichtlinie stammen aus der bereitgestellten Datei `penjpa-objectGridDeployment.xml`.
3. **Konfiguration der ObjectGrid-Clientkorrekturwerte**: Verwenden Sie die Datei `META-INF/openjpa-objectGrid-client-override.xml`. Diese Datei wird verwendet, um ein clientseitiges ObjectGrid anzupassen. Standardmäßig wendet der ObjectGrid-Cache eine ObjectGrid-Standardkonfiguration für Clientkorrekturwerte an, die einen nahen Cache inaktiviert. Wenn eine Anwendung einen nahen Cache erfordert, können Sie diese Datei bereitstellen und `numberOfBuckets="xxx"` angeben. Der Standardclientkorrekturwert inaktiviert den nahen Cache mit `numberOfBuckets="0"`. Der nahe Cache kann aktiv sein, wenn das Attribut "numberOfBuckets" über die Datei `openjpa-objectGrid-client-override.xml` auf einen Wert größer als 0 zurückgesetzt wird. Die Datei `openjpa-objectGrid-client-override.xml` funktioniert ähnlich wie die Datei `openjpa-objectGrid.xml`. Sie überscheibt oder erweitert die Standardkonfiguration für ObjectGrid-Clientkorrekturwerte.

Beispiele für OpenJPA-ObjectGrid-XML-Dateien

OpenJPA-ObjectGrid-XML-Dateien müssen auf der Basis der Persistenzeinheit erstellt werden.

Im Folgenden sehen Sie eine Beispieldatei `persistence.xml`, die die Konfiguration einer Persistenzeinheit darstellt:

persistence.xml

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  version="1.0">
  <persistence-unit name="AnnuityGrid">
    <provider>org.apache.openjpa.persistence.PersistenceProviderImpl</provider>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.FixedAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.EquityAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout</class>
```

```
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Person</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityHolder</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Address</class>
    <exclude-unlisted-classes>true</exclude-unlisted-classes>

    <properties>
    <!-- Database setting -->


    <!--  enable cache -->
      <property name="openjpa.DataCache"
        value="com.ibm.websphere.objectgrid.openjpa.ObjectGridDataCache(objectGridName=Annuity,
              objectGridType=EMBEDDED, maxNumberOfReplicas=4)" />
      <property name="openjpa.RemoteCommitProvider" value="sjvm" />
      <property name="openjpa.QueryCache"
                value="com.ibm.websphere.objectgrid.openjpa.ObjectGridQueryCache()" />
    </properties>
  </persistence-unit>
</persistence>
```

Es folgt die Datei `openjpa-objectGrid.xml`, die der Datei `persistence.xml` entspricht:

openjpa-objectGrid.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="Annuity">
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject"
                  readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
      <backingMap name="ObjectGridQueryCache" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" pluginCollectionRef="ObjectGridQueryCache"
                  evictionTriggers="MEMORY_USAGE_THRESHOLD"  />
    </objectGrid>
  </objectGrids>
  <backingMapPluginCollections>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
```

```
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection
            id="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="ObjectGridQueryCache">
      <bean id="MapIndexPlugin"  className="com.ibm.websphere.objectgrid.plugins.index.HashIndex">
          <property name="Name" type="java.lang.String"
                    value="QueryCacheKeyIndex" description="name of index"/>
          <property name="POJOKeyIndex" type="boolean" value="true" description="POJO Key Index "/>
      </bean>
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Anmerkung:

1. Jede Entität wird einer BackingMap zugeordnet, die denselben Namen wie die vollständig qualifizierte Entitätsklasse hat.
2. Wenn Entitätsklassen in einer Vererbungshierarchie enthalten sind, werden untergeordnete Klassen der BackingMap der übergeordneten Klasse zugeordnet. Das bedeutet, dass eine Vererbungshierarchie eine einzige BackingMap nutzt.
3. Die ObjectGridQueryCache-Map ist für die Unterstützung von QueryCache erforderlich.
4. Die backingMapPluginCollection für jede Entitäts-Map erfordert, dass der ObjectTransformer die Klasse "com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" verwendet.
5. Die backingMapPluginCollection für ObjectGridQueryCache muss einen Schlüsselindex mit dem Namen "QueryCacheKeyIndex" haben, wie im folgenden Beispiel gezeigt wird.
6. Der Evictor ist für jede Map optional.

Es folgt die Datei `openjpa-objectGridDeployment.xml`, die der Datei `persistence.xml` entspricht:

openjpa-objectGridDeployment.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="Annuity">
    <mapSet name="MAPSET_Annuity" numberOfPartitions="1" numInitialContainers="1"
            minSyncReplicas="0" maxSyncReplicas="4" maxAsyncReplicas="0"
            replicaReadEnabled="true">
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
```

```
      <map ref="ObjectGridQueryCache" />
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

Anmerkung: Die ObjectGridQueryCache-Map ist für die Unterstützung von QueryCache erforderlich.

Externes System für einen Cache mit dem ObjectGrid-Typ REMOTE

Sie müssen ein externes System einrichten, wenn Sie einen Cache mit dem ObjectGrid-Typ REMOTE konfigurieren möchten. Sie benötigen ObjectGrid- und ObjectGridDeployment-XML-Konfigurationsdateien, die auf einer Datei `persistence.xml` basieren, um ein externes System einrichten zu können. Die OpenJPA-ObjectGrid- und -ObjectGridDeployment-XML-Konfigurationsdateien, die im Abschnitt mit den Beispielen für die OpenJPA-ObjectGrid-XML-Dateien beschrieben werden, können auch zum Einrichten eines externen eXtreme-Scale-Systems verwendet werden.

Ein externes eXtreme-Scale-System hat sowohl Katalogservice- als auch Containerserverprozesse. Sie müssen den Katalogserver starten, bevor Sie die Containerserver starten.

Fehlerbehebung

1. **CacheException: Failed to get ObjectGrid server**

   Bei einem ObjectGrid des Typs EMBEDDED oder EMBEDDED_PARTITION versucht der eXtreme-Scale-Cache, eine Serverinstanz von der Laufzeitumgebung abzurufen. In einer Java-SE-Umgebung wird ein Server von eXtreme Scale mit integriertem Katalogservice gestartet. Der integrierte Katalogservice versucht, an Port 2809 empfangsbereit zu sein. Wenn dieser Port von einem anderen Prozess verwendet wird, tritt dieser Fehler auf. Wenn externe Katalogserviceendpunkte angegeben werden, z. B. in der Datei `objectGridServer.properties`, tritt dieser Fehler auf, wenn der Hostname oder Port falsch angegeben sind.

2. **CacheException: Failed to get REMOTE ObjectGrid for configured REMOTE ObjectGrid. objectGridName = [ObjectGridName], PU name = [persistenceUnitName]**

   Dieser Fehler tritt auf, wenn der Cache kein ObjectGrid von den bereitgestellten Katalogserviceendpunkten abrufen kann. Gewöhnlich ist dieser Fehler auf einen falsch angegebenen Hostnamen oder Port zurückzuführen.

3. **CacheException: Cannot have two PUs [persistenceUnitName_1, persistenceUnitName_2] configured with same ObjectGridName [ObjectGridName] of EMBEDDED ObjectGridType**

   Diese Ausnahme tritt ein, wenn Sie viele Persistenzeinheitenkonfigurationen haben und die eXtreme-Scale-Caches dieser Einheiten mit demselben ObjectGrid-Namen und dem ObjectGrid-Typ EMBEDDED konfiguriert sind. Diese Persistenzeinheitenkonfigurationen können in derselben oder in unterschiedlichen Dateien `persistence.xml` enthalten sein. Sie müssen sicherstellen, dass der ObjectGrid-Name für jede Persistenzeinheit eindeutig ist, wenn der ObjectGrid-Typ EMBEDDED verwendet wird.

4. **CacheException: REMOTE ObjectGrid [ObjectGridName] does not include required BackingMaps [mapName_1, mapName_2,...]**

   Wenn der ObjectGrid-Typ REMOTE verwendet wird und das abgerufene clientseitige ObjectGrid keine vollständigen Entitäts-BackingMaps für die Unterstützung des Caches der Persistenzeinheit hat, wird diese Ausnahme ausgelöst. Beispiel: Es sind fünf Entitätsklassen in der Konfiguration der Persistenzeinheit

aufgelistet, aber das abgerufene ObjectGrid hat nur zwei BackingMaps. Diese Ausnahme wird auch dann ausgelöst, wenn das abgerufene ObjectGrid zehn BackingMaps enthält, aber eine der fünf erforderlichen Entitäts-BackingMaps nicht unter den zehn vorhandenen gefunden wird.

Anmerkung: Das Datenformat des eXtreme-Scale-Caches für OpenJPA wurde geändert, um eine bessere Leistung zu erzielen. Alle Systeme, auf denen OpenJPA-Anwendungen ausgeführt werden, die mit eXtreme Scale als L2-Cache konfiguriert sind, müssen gestoppt werden, bevor die Migration auf WebSphere eXtreme Scale Version 7.0 durchgeführt wird.

HTTP-Sitzungsmanager konfigurieren

Der HTTP-Sitzungsmanager stellt Sitzungsreplikationsfunktionen für eine zugeordnete Anwendung bereit. Der Sitzungsreplikationsmanager arbeitet mit dem Webcontainer für den Sitzungsmanager zusammen, um HTTP-Sitzungen zu erstellen und die Lebenszyklen von HTTP-Sitzungen zu verwalten, die der Anwendung zugeordnet sind.

XML-Dateien für die Konfiguration des HTTP-Sitzungsmanagers

Wenn Sie einen Containerserver starten, der HTTP-Sitzungsdaten speichert, können Sie die Standard-XML-Dateien verwenden, oder Sie können angepasste XML-Dateien verwenden, die bestimmte ObjectGrid-Namen, die Anzahl der Replikate usw. festlegen.

Position der Musterdateien

Diese XML-Dateien sind in "<*extremescale*>/ObjectGrid/session/samples" für eine eigenständige Installation bzw. "<*WAS-Installationsstammverzeichnis*>/optionalLibraries/ObjectGrid/session/samples" für eine Installation von WebSphere eXtreme Scale in einer Zelle von WebSphere Application Server gepackt.

Integriertes XML-Paket

Wenn Sie ein integriertes Szenario konfigurieren, was bedeutet, dass der Containerserver auf der Webcontainerschicht gestartet wird, werden die Dateien "objectGrid.xml" und "objectGridDeployment.xml" standardmäßig bereitgestellt. Sie können diese Dateien aktualisieren, um das Verhalten des HTTP-Sitzungsmanagers anzupassen.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd" xmlns="http://ibm.com/ws/objectgrid/config">
 <objectGrids>
  <objectGrid name="session">
   <bean id="ObjectGridEventListener" className="com.ibm.ws.xs.sessionmanager.SessionHandleManager"/>
   <backingMap name="objectgridSessionMetadata" pluginCollectionRef="objectgridSessionMetadata" readOnly="false"
     lockStrategy="PESSIMISTIC" ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3600" copyMode="NO_COPY"/>
   <backingMap name="objectgridSessionAttribute.*" template="true" readOnly="false" lockStrategy="PESSIMISTIC"
     ttlEvictorType="NONE" copyMode="NO_COPY"/>
       <backingMap name="objectgridSessionTTL.*" template="true" readOnly="false" lockStrategy="PESSIMISTIC"
     ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3600" copyMode="NO_COPY"/>
  </objectGrid>
 </objectGrids>
    <backingMapPluginCollections>
        <backingMapPluginCollection id="objectgridSessionMetadata">
            <bean id="MapEventListener" className="com.ibm.ws.xs.sessionmanager.MetadataMapListener"/>
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

Werte, die Sie ändern können:

- Attribut **ObjectGrid name**: Dieses Attribut muss mit der Eigenschaft "objectGridName" in der Datei `splicer.properties`, die verwendet wird, um die Webanwendung zusammenzufügen, und dem Attribut "objectgridName" in der Datei `objectGridDeployment.xml` übereinstimmen. Wenn Sie mehrere Anwendungen haben und möchten, dass die Sitzungsdaten in unterschiedlichen Grids gespeichert werden, müssen diese Anwendungen unterschiedliche Wert für das Attribut "ObjectGrid name" haben. Der Name des ObjectGrids ist die einzige Information, die in dieser Datei geändert werden kann.

Werte, die Sie nicht ändern können:

- **ObjectGridEventListener:** Die ObjectGridEventListener-Zeile kann nicht geändert werden und wird intern verwendet.
- **objectgridSessionMetadata**: Die objectgridSessionMetadata-Zeile verweist auf die Map, in der die Metadaten der HTTP-Sitzung gespeichert sind. Es gibt einen Eintrag für jede in diesem Grid gespeicherte HTTP-Sitzung in dieser Map.
- **objectgridSessionAttribute.*** Die objectgridSessionAttribute.*-Zeile definiert eine dynamische Map, die verwendet wird, um die Map zu erstellen, in der HTTP-Sitzungsattribute gespeichert werden, wenn der Parameter **fragmentedSession** in der Datei `splicer.properties`, die für die Zusammenfügung der Webanwendung verwendet wird, auf `true` gesetzt ist. Diese dynamische Map hat den Namen "objectgridSessionAttribute". Auf der Basis dieser Schablone wird eine weitere Map mit dem Namen "objectgridSessionAttributeEvicted" erstellt, in der Sitzungen gespeichert werden, die das zulässige Zeitlimit überschritten haben, aber vom Webcontainer nicht ungültig gemacht wurden.
- Die **MetadataMapListener**-Zeile ist intern und kann nicht geändert werden.

Abbildung 14. Datei "objectGrid.xml"

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

 <objectgridDeployment objectgridName="session">
  <mapSet name="sessionMapSet" numberOfPartitions="5" minSyncReplicas="0" maxSyncReplicas="0"
    maxAsyncReplicas="1" developmentMode="false" placementStrategy="PER_CONTAINER">
             <map ref="objectgridSessionMetadata"/>
             <map ref="objectgridSessionAttribute.*"/>
             <map ref="objectgridSessionTTL.*"/>
    </mapSet>
   </objectgridDeployment>
</deploymentPolicy>
```

Werte, die Sie ändern können:

- Attribut **objectgridName**: Dieses Attribut muss mit der Eigenschaft "objectGridName" in der Datei `splicer.properties`, die für das Zusammenfügen der Webanwendung verwendet wird, und mit dem Attribut "ObjectGrid name" in der Datei `objectGrid.xml` übereinstimmen.
- Attribute des Elements **mapSet**: Sie können alle mapSet-Eigenschaften mit Ausnahme des Attributs "placementStrategy" ändern.

  Name Kann in jeden Wert aktualisiert werden.

  numberOfPartitions
  Gibt die Anzahl primärer Partitionen an, die in jedem Server gestartet werden, der die Webanwendung enthält. Wenn Sie Partitionen hinzufügen, werden die Daten für den Fall eines Failovers breiter verteilt. Der Standardwert sind 5 Partitionen und reicht für die meisten Anwendungen aus.

  minSyncReplicas, maxSyncReplicas und maxAsyncReplicas
  Gibt die Anzahl und den Typ der Replikate an, in denen die HTTP-Sitzungsdaten gespeichert werden. Standardmäßig wird 1 asynchrones Replikat verwendet, das für die meisten Anwendungen ausreicht. Synchrone Replikation findet im Anforderungspfad statt, was die Antwortzeiten für die Webanwendung erhöhen kann.

  developmentMode
  Informiert den eXtreme-Scale-Verteilungsservice darüber, ob die Replikat-Shards für eine Partition auf demselben Knoten wie das primäre Shards verteilt werden können. Sie können den Wert in einer Entwicklungsumgebung auf "true" setzen, sollten diese Funktion in einer Produktionsumgebung aber inaktivieren, weil der Ausfall eines Knotens zum Verlust von Sitzungsdaten führen könnte.

  placementStrategy
  Ändern Sie den Wert dieses Attributs nicht.
- Der Rest der Datei verweist auf dieselben Map-Namen, die auch in der Datei "objectGrid.xml" enthalten sind. Diese Namen können nicht geändert werden.

Werte, die Sie nicht ändern können:

- Attribut "placementStrategy" im Element "mapSet"

Abbildung 15. Datei "objectGridDeployment.xml"

Fernes XML-Paket

Wenn Sie den Fernmodus verwenden, in dem die Container als eigenständige Prozesse ausgeführt werden, müssen Sie die Dateien `objectGridStandAlone.xml` und `objectGridDeploymentStandAlone.xml` verwenden, um die Prozesse zu starten. Sie können diese Dateien aktualisieren, um die Konfiguration zu ändern.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
 <objectGrids>
  <objectGrid name="session">
   <bean id="ObjectGridEventListener" className="com.ibm.ws.xs.sessionmanager.SessionHandleManager"/>
          <backingMap name="objectgridSessionMetadata" pluginCollectionRef="objectgridSessionMetadata"
       readOnly="false" lockStrategy="PESSIMISTIC" ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3600"
       copyMode="COPY_TO_BYTES"/>
          <backingMap name="objectgridSessionAttribute.*" template="true" readOnly="false" lockStrategy="PESSIMISTIC"
       ttlEvictorType="NONE" copyMode="COPY_TO_BYTES"/>
          <backingMap name="objectgridSessionTTL.*" template="true" readOnly="false" lockStrategy="PESSIMISTIC"
       ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3600" copyMode="COPY_TO_BYTES"/>
  </objectGrid>
 </objectGrids>
    <backingMapPluginCollections>
        <backingMapPluginCollection id="objectgridSessionMetadata">
            <bean id="MapEventListener" className="com.ibm.ws.xs.sessionmanager.MetadataMapListener"/>
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

Werte, die Sie ändern können:

- Attribut **objectgridName**: Dieses Attribut kann geändert werden, muss aber mit der Eigenschaft "objectGridName" in der Datei `splicer.properties`, die zum Zusammenfügen der Webanwendung verwendet wird, und mit dem Attribut "objectgridName" in der Datei `objectGridDeploymentStandAlone.xml` übereinstimmen. Wenn Sie mehrere Anwendungen haben und möchten, dass die Sitzungsdaten in unterschiedlichen Grids gespeichert werden, müssen diese Anwendungen unterschiedliche Grid-Namen haben. Der Name des Grids ist die einzige Information, die in dieser Datei geändert werden kann.

Werte, die Sie nicht ändern können:

- **ObjectGridEventListener:** Die ObjectGridEventListener-Zeile kann nicht geändert werden und wird intern verwendet.
- **objectgridSessionMetadata**: Die objectgridSessionMetadata-Zeile verweist auf die Map, in der die Metadaten der HTTP-Sitzung gespeichert sind. Es gibt einen Eintrag für jede in diesem Grid gespeicherte HTTP-Sitzung in dieser Map.
- **objectgridSessionAttribute.*** Die objectgridSessionAttribute.*-Zeile definiert eine dynamische Map, die verwendet wird, um die Map zu erstellen, in der HTTP-Sitzungsattribute gespeichert werden, wenn der Parameter **fragmentedSession** in der Datei `splicer.properties`, die für die Zusammenfügung der Webanwendung verwendet wird, auf `true` gesetzt ist. Diese dynamische Map hat den Namen "objectgridSessionAttribute". Auf der Basis dieser Schablone wird eine weitere Map mit dem Namen "objectgridSessionAttributeEvicted" erstellt, in der Sitzungen gespeichert werden, die das zulässige Zeitlimit überschritten haben, aber vom Webcontainer nicht ungültig gemacht wurden.
- Die **MetadataMapListener**-Zeile ist intern und kann nicht geändert werden.

Abbildung 16. objectGridStandAlone.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

 <objectgridDeployment objectgridName="session">
  <mapSet name="sessionMapSet" numberOfPartitions="5" minSyncReplicas="0" maxSyncReplicas="0"
   maxAsyncReplicas="1" developmentMode="false" placementStrategy="PER_CONTAINER">
            <map ref="objectgridSessionMetadata"/>
            <map ref="objectgridSessionAttribute.*"/>
            <map ref="objectgridSessionTTL.*"/>
     </mapSet>
    </objectgridDeployment>
</deploymentPolicy>
```

Werte, die Sie ändern können:

- Attribut **objectgridName**: Dieses Attribut muss mit der Eigenschaft "objectGridName" in der Datei `splicer.properties`, die für das Zusammenfügen der Webanwendung verwendet wird, und mit dem Attribut "ObjectGrid name" in der Datei `objectGrid.xml` übereinstimmen.
- Attribute des Elements **mapSet**: Sie können alle mapSet-Eigenschaften mit Ausnahme des Attributs "placementStrategy" ändern.

  Name Kann in jeden Wert aktualisiert werden.

  numberOfPartitions
  Gibt die Anzahl primärer Partitionen an, die in jedem Server gestartet werden, der die Webanwendung enthält. Wenn Sie Partitionen hinzufügen, werden die Daten für den Fall eines Failovers breiter verteilt. Der Standardwert sind 5 Partitionen und reicht für die meisten Anwendungen aus.

  minSyncReplicas, maxSyncReplicas und maxAsyncReplicas
  Gibt die Anzahl und den Typ der Replikate an, in denen die HTTP-Sitzungsdaten gespeichert werden. Standardmäßig wird 1 asynchrones Replikat verwendet, das für die meisten Anwendungen ausreicht. Synchrone Replikation findet im Anforderungspfad statt, was die Antwortzeiten für die Webanwendung erhöhen kann.

  developmentMode
  Informiert den eXtreme-Scale-Verteilungsservice darüber, ob die Replikat-Shards für eine Partition auf demselben Knoten wie das primäre Shards verteilt werden können. Sie können den Wert in einer Entwicklungsumgebung auf "true" setzen, sollten diese Funktion in einer Produktionsumgebung aber inaktivieren, weil der Ausfall eines Knotens zum Verlust von Sitzungsdaten führen könnte.

  placementStrategy
  Ändern Sie den Wert dieses Attributs nicht.
- Der Rest der Datei verweist auf dieselben Map-Namen, die auch in der Datei `objectGrid.xml` enthalten sind. Diese Namen können nicht geändert werden.

Werte, die Sie nicht ändern können:

- Attribut "placementStrategy" im Element "mapSet"

Abbildung 17. objectGridDeploymentStandAlone.xml

HTTP-Sitzungsmanager mit WebSphere Application Server konfigurieren

WebSphere Application Server stellt zwar Funktionen für das Sitzungsmanagement bereit, aber diese Unterstützung ist unter sehr hohen Anforderungslasten nicht skalierbar. Im Lieferumfang von WebSphere eXtreme Scale ist eine Sitzungsmanagementimplementierung enthalten, die Sitzungsreplikation, hohe Verfügbarkeit, eine bessere Skalierbarkeit und stabilere Konfigurationsoptionen bietet.

Vorbereitende Schritte

- WebSphere eXtreme Scale muss in der Zelle mit WebSphere Application Server oder WebSphere Application Server Network Deployment installiert werden, damit der Sitzungsmanager von eXtreme Scale verwendet wird. Weitere Informationen finden Sie im Abschnitt „WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client mit WebSphere Application Server integrieren" auf Seite 23.

Informationen zu diesem Vorgang

Der HTTP-Sitzungsmanager von WebSphere eXtreme Scale unterstützt integrierte und ferne Server für das Caching.

Szenario mit integriertem Sitzungsmanager

Im Szenario mit dem integrierten Sitzungsmanager werden die WebSphere eXtreme Scale-Server in denselben Prozessen wie die Servlets ausgeführt. Der Sitzungsmanager kann direkt mit der lokalen ObjectGrid-Instanz kommunizieren, wodurch teure Verzögerungen bei der Netzübertragung vermieden werden.

Wenn Sie WebSphere Application Server verwenden, kopieren Sie die mitgelieferten Dateien `ObjectGrid-Stammverzeichnis/session/samples/objectGrid.xml` und `ObjectGrid-Stammverzeichnis/session/samples/objectGridDeployment.xml` in die `META-INF`-Verzeichnisse Ihrer WAR-Dateien. eXtreme Scale erkennt diese Dateien beim Anwendungsstart automatisch und startet automatisch die eXtreme-Scale-Container in demselben Prozess wie den Sitzungsmanager.

Sie können die Datei `objectGridDeployment.xml` ändern, abhängig davon, ob Sie synchrone oder asynchrone Replikation verwenden möchten und wie viele Replikate konfiguriert werden sollen.

Szenario mit fernen Servern

Im Szenario mit fernen Servern werden die Containerserver in anderen Prozessen als die Servlets ausgeführt. Der Sitzungsmanager kommuniziert mit einem fernen Containerserver. Wenn Sie einen fernen, über ein Netz verbundenen Containerserver verwenden möchten, muss der Sitzungsmanager mit den Hostnamen und Portnummern der Katalogservicedomäne konfiguriert werden. Der Sitzungsmanager verwendet anschließend eine eXtreme-Scale-Clientverbindung, um mit dem Katalogserver und den Containerservern zu kommunizieren.

Wenn die Containerserver in unabhängigen, eigenständigen Prozessen gestartet werden, starten Sie die eXtreme-Scale-Container mit den Dateien `objectGridStandAlone.xml` und `objectGridDeploymentStandAlone.xml`, die im Verzeichnis "samples" des Sitzungsmanagers bereitgestellt werden.

Vorgehensweise

1. Fügen Sie Ihre Anwendung so zusammen, dass sie den Sitzungsmanager verwenden kann. Wenn Sie den Sitzungsmanager verwenden möchten, müssen Sie die entsprechenden Filterdeklarationen den Webimplementierungsdeskriptoren für die Anwendung hinzufügen. Außerdem werden die Konfigurationsparameter des Sitzungsmanagers in Form von Initialisierungsparametern für den Servlet-Kontext in den Implementierungsdeskriptoren an den Sitzungsmanager übergeben. Es gibt mehrere Methoden, mit denen Sie diese Informationen in Ihre Anwendung einführen können:

   - **7.1+ In der Administrationskonsole von WebSphere Application Server**

Sie können die Verwendung des HTTP-Sitzungsmanager von WebSphere eXtreme Scale beim Installieren Ihrer Anwendung konfigurieren. Die Anwendungs- bzw. Serverkonfiguration kann auch geändert werden, damit der HTTP-Sitzungsmanager von WebSphere eXtreme Scale verwendet wird. Weitere Informationen finden Sie im Abschnitt „Sitzungspersistenz für ein Daten-Grid erstellen” auf Seite 267.

Anmerkung: Diese Option funktioniert nur, wenn alle Knoten, auf denen die Anwendung ausgeführt wird, die Datei `splicer.properties` in demselben Pfad enthalten. Für heterogene Umgebungen, die Windows- und UNIX-Knoten enthalten, ist diese Option nicht möglich, und Sie müssen die Anwendung manuell zusammenfügen.

- **Option für automatische Zusammenfügung**

  Sie müssen Ihre Anwendungen nicht manuell zusammenfügen, wenn die Anwendung in WebSphere Application Server oder WebSphere Application Server Network Deployment ausgeführt wird. Sie können einer Zelle oder einem Server eine angepasste Eigenschaft hinzufügen, die sich auf alle Webanwendungen dieser Ebene auswirkt. Die Eigenschaft hat den Namen `com.ibm.websphere.xs.sessionFilterProps` und muss auf die Position der Datei `splicer.properties` gesetzt werden, die Ihre Anwendungen erfordern.

  Im Folgenden sehen Sie einen Beispielpfad für die Position der Datei: `/opt/splicer.properties`.

  – **Verwenden Sie die Administrationskonsole, um alle Webwendungen in einer Zelle als zusammengefügt anzugeben:**

    Klicken Sie auf **Systemverwaltung** → **Cell** → **Angepasste Eigenschaften**, und fügen Sie die Eigenschaft hinzu.

  – **Verwenden Sie die Administrationskonsole, um alle Webanwendungen in einem bestimmten Anwendungsserver anzugeben:**

    Klicken Sie auf **Anwendungsserver** → **<*Servername*>** → **Verwaltung** → **Angepasste Eigenschaften**, und fügen Sie die Eigenschaft hinzu.

  Die einzigen gültigen Geltungsbereiche sind "Zelle" und "Server" und nur verfügbar, wenn mit einem Deployment Manager gearbeitet wird. Wenn Sie einen anderen Geltungsbereich benötigen, müssen Sie Ihre Webanwendungen manuell zusammenfügen.

  Anmerkung: Die Option für automatische Zusammenfügung funktioniert nur, wenn alle Knoten, auf denen die Anwendung ausgeführt wird, die Datei `splicer.properties` in demselben Pfad enthalten. Für heterogene Umgebungen, die Windows- und UNIX-Knoten enthalten, ist diese Option nicht möglich, und Sie müssen die Anwendung manuell zusammenfügen.

- **Script "addObjectGridFilter"**

  Verwenden Sie ein Befehlszeilenscript, das mit eXtreme Scale bereitgestellt wird, um eine Anwendung mit Filterdeklarationen und Konfigurationen in Form von Initialisierungsparametern für den Servlet-Kontext zusammenzufügen. Dieses Script, das den Namen `ObjectGrid-Stammverzeichnis/bin/addObjectGridFilter.sh` bzw. `ObjectGrid-Stammverzeichnis/session/bin/addObjectGridFilter.bat` hat, akzeptiert zwei Parameter: die Anwendung (absoluter Pfad der EAR-Datei), die zusammengefügt werden muss, und den absoluten Pfad der Eigenschaftendatei, die die verschiedenen Konfigurationsparameter enthält. Das Syntaxformat dieses Scripts ist wie folgt:

  Windows

  ```
  addObjectGridFilter.bat [Position der EAR-Datei] [Position der Eigenschaftendatei]
  ```

UNIX

```
addObjectGridFilter.sh [Position der EAR-Datei] [Position der Eigenschaftendatei]
```

UNIX **Beispiel mit einer Installation von eXtreme Scale in WebSphere Application Server unter UNIX:**

a. `cd ObjectGrid-Stammverzeichnis/optionalLibraries/ObjectGrid/session/bin`

b. `addObjectGridFilter.sh /tmp/mySessionTest.ear WAS-Stammverzeichnis/optionalLibraries/ObjectGrid/session/samples/splicer.properties`

UNIX **Beispiel mit einer Installation von eXtreme Scale in einem eigenständigen Verzeichnis unter UNIX:**

a. `cd ObjectGrid-Stammverzeichnis/session/bin`

b. `addObjectGridFilter.sh /tmp/mySessionTest.ear ObjectGrid-Stammverzeichnis/session/samples/splicer.properties`

Der Servlet-Filter, der eingefügt wird, verwaltet die Standardwerte für Konfigurationswerte. Sie können diese Standardwerte mit Konfigurationsoptionen überschreiben, die Sie in der Eigenschaftendatei im zweiten Argument angeben. Eine Liste der Parameter, die Sie verwenden können, finden Sie im Abschnitt „Initialisierungsparameter für den Servlet-Kontext" auf Seite 274.

Sie können die Musterdatei `splicer.properties` ändern und verwenden, die mit der Installation von eXtreme Scale bereitgestellt wird. Sie können auch das Script "addObjectGridServlets" verwenden, das den Sitzungsmanager einfügt, indem Sie jedes Servlet erweitern. Das empfohlene Script ist jedoch das Script "addObjectGridFilter".

- **Ant-Build-Script**

  Im Lieferumfang von WebSphere eXtreme Scale ist eine Datei `build.xml` enthalten, das von Apache Ant verwendet werden kann, und im Ordner `WAS-Stammverzeichnis/bin` einer Installation von WebSphere Application Server enthalten ist. Die Datei `build.xml` kann bearbeitet werden, um die Konfigurationseigenschaften des Sitzungsmanagers zu ändern. Die Konfigurationseigenschaften sind identisch mit den Eigenschaftsnamen in der Datei `splicer.properties`. Rufen Sie nach der Änderung der Datei `build.xml` den Ant-Prozess mit den folgenden Scripts auf:

  – UNIX `ant.sh`, `ws_ant.sh`

  – Windows `ant.bat`, `ws_ant.bat`

  (UNIX) bzw. (Windows).

- **Webdeskriptor manuell aktualisieren**

  Editieren Sie die mit der Webanwendung gepackte Datei `web.xml`, um die Filterdeklaration, die Servlet-Zuordnung und die Initialisierungsparameter für den Servlet-Kontext zu integrieren. Sie sollten diese Methode nicht verwenden, weil sie fehleranfällig ist.

Eine Liste der Parameter, die Sie verwenden können, finden Sie im Abschnitt „Initialisierungsparameter für den Servlet-Kontext" auf Seite 274.

2. Implementieren Sie die Anwendung. Führen Sie dazu die Schritte aus, die Sie gewöhnlich für einen Server oder Cluster verwenden. Nach der Implementierung der Anwendung können Sie die Anwendung starten.
3. Greifen Sie auf die Anwendung zu. Sie können jetzt auf die Anwendung zugreifen, die mit dem Sitzungsmanager und mit WebSphere eXtreme Scale interagiert.

Nächste Schritte

Sie können die meisten Konfigurationsparameter des Sitzungsmanagers ändern, wenn Sie Ihre Anwendung für die Verwendung des Sitzungsmanagers instrumentieren. Zu diesen Attributen gehören die synchrone oder asynchrone Replikation, die Länge der Sitzungs-ID, die Größe der speicherinternen Sitzungstabelle usw. Abgesehen von den Attributen, die während der Instrumentierung der Anwendung geändert werden können, sind die einzigen Attribute, die Sie nach der Anwendungsimplementierung ändern können, die Attribute, die sich auf die WebSphere eXtreme Scale-Serverclustertopologie beziehen, und die Art und Weise, in der Clients (Sitzungsmanager) eine Verbindung zu diesen Servern herstellen.

Sitzungspersistenz für ein Daten-Grid erstellen:

Sie können Ihre Anwendung von WebSphere Application Server für die persistente Speicherung von Sitzungen in einem Daten-Grid konfigurieren. Dieses Daten-Grid kann in einem integrierten Containerserver, der in WebSphere Application Server ausgeführt wird, oder in einem fernen Daten-Grid enthalten sein.

Vorbereitende Schritte

Bevor Sie die Konfiguration in WebSphere Application Server ändern, müssen Sie die folgenden Informationen zusammenstellen:

- Name des Sitzungsdaten-Grids, das Sie verwenden möchten. Weitere Informationen zum Erstellen eines Sitzungsdaten-Grids finden Sie unter „HTTP-Sitzungsmanager mit WebSphere Application Server konfigurieren" auf Seite 263.
- Wenn der Katalogservice, den Sie für die Verwaltung Ihrer Sitzungen verwenden möchten, außerhalb der Zelle befindet, in der Sie Ihre Sitzungsanwendung installieren, müssen Sie eine Katalogservicedomäne erstellen. Weitere Informationen finden Sie unter „Katalogservicedomänen in WebSphere Application Server erstellen" auf Seite 350.

Vorgehensweise

- **Gehen Sie wie folgt vor, um das Sitzungsmanagement bei der Installation der Anwendung zu konfigurieren:**
  1. Klicken Sie in der Administrationskonsole von WebSphere Application Server auf **Anwendungen** → **Neue Anwendung** → **Neue Unternehmensanwendung**. Wählen Sie den Pfad **Detailliert** für die Erstellung der Anwendung aus, und führen Sie die ersten Schritte im Assistenten aus.
  2. Konfigurieren Sie im Schritt **eXtreme Scale - Einstellungen für das Sitzungsmanagement** das Daten-Grid, das Sie verwenden möchten. Wählen Sie **Fernes eXtreme-Scale-Daten-Grid** oder **Integriertes eXtreme-Scale-Daten-Grid** aus.
     - Für die Option **Fernes eXtreme-Scale-Daten-Grid** wählen Sie die Katalogservicedomäne aus, die das Sitzungsdaten-Grid verwaltet. Wählen Sie anschließend in der Liste aktiver Sitzungsdaten-Grids ein Daten-Grid aus.
     - Für die Option **Integriertes eXtreme-Scale-Daten-Grid** wählen Sie die ObjectGrid-Standardkonfiguration aus, oder Sie geben Sie Position Ihrer ObjectGrid-Konfigurationsdateien an.
  3. Führen Sie die Schritte im Assistenten aus, um Ihre Anwendung zu installieren.

  Sie können die Anwendung auch mit einem wsadmin-Script installieren. Im folgenden Beispiel erstellt der Parameter **-SessionManagement** dieselbe Konfiguration, die Sie auch über die Administrationskonsole erstellen können:

Für die Konfiguration eines fernen eXtreme-Scale-Daten-Grids:

```
AdminApp.install('C:/A.ear', '[  -nopreCompileJSPs -distributeApp
-nouseMetaDataFromBinary -nodeployejb -appname A -edition 8.0
-createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
off -noprocessEmbeddedConfig -filepermission .*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755
-buildVersion Unknown -noallowDispatchRemoteInclude -noallowServiceRemoteInclude
-asyncRequestDispatchType DISABLED -nouseAutoLink -SessionManagement [[true
XSRemoteSessionManagement cs0:!:grid0]]
-MapWebModToVH [[MicroWebApp microwebapp.war,WEB-INF/web.xml default_host] [MicroSipApp
microsipapp.war,WEB-INF/web.xml default_host] [MicroDG1App microdg1app.war,WEB-INF/web.xml
default_host] [MicroDG2App microdg2app.war,WEB-INF/web.xml default_host] [MicroSip2App
microsip2app.war,WEB-INF/web.xml default_host]]]')
```

Für das integrierte eXtreme-Scale-Szenario mit Standardkonfiguration:

```
AdminApp.install('C:/A.ear', '[  -nopreCompileJSPs -distributeApp
-nouseMetaDataFromBinary -nodeployejb -appname A -edition 8.0
-createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
off -noprocessEmbeddedConfig -filepermission .*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755
-buildVersion Unknown -noallowDispatchRemoteInclude -noallowServiceRemoteInclude
-asyncRequestDispatchType DISABLED -nouseAutoLink -SessionManagement [[true
XSRemoteSessionManagement  :!: :!:default]]  -MapWebModToVH [[MicroWebApp microwebapp.war,
WEB-INF/web.xml default_host] [MicroSipApp
microsipapp.war,WEB-INF/web.xml default_host] [MicroDG1App microdg1app.war,WEB-INF/web.xml
default_host] [MicroDG2App microdg2app.war,WEB-INF/web.xml default_host] [MicroSip2App
microsip2app.war,WEB-INF/web.xml default_host]]]')
```

Für das integrierte eXtreme-Scale-Szenario mit einer angepassten Konfiguration:

```
AdminApp.install('C:/A.ear', '[  -nopreCompileJSPs -distributeApp
-nouseMetaDataFromBinary -nodeployejb -appname A -edition 8.0
-createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
off -noprocessEmbeddedConfig -filepermission .*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755
-buildVersion Unknown -noallowDispatchRemoteInclude -noallowServiceRemoteInclude
-asyncRequestDispatchType DISABLED -nouseAutoLink -SessionManagement [[true
XSRemoteSessionManagement  :!: :!:custom:!:c:\XS\objectgrid.xml:!:c:\XS\objectgriddeployment.xml]]
-MapWebModToVH [[MicroWebApp microwebapp.war,WEB-INF/web.xml default_host] [MicroSipApp
microsipapp.war,WEB-INF/web.xml default_host] [MicroDG1App microdg1app.war,WEB-INF/web.xml
default_host] [MicroDG2App microdg2app.war,WEB-INF/web.xml default_host] [MicroSip2App
microsip2app.war,WEB-INF/web.xml default_host]]]')
```

- **Gehen Sie wie folgt vor, um das Sitzungsmanagement in einer vorhandenen Anwendung über die Administrationskonsole von WebSphere Application Server zu konfigurieren:**

  1. Klicken Sie in der Administrationskonsole von WebSphere Application Server auf **Anwendungen** → **Anwendungstypen** → **WebSphere-Unternehmensanwendungen** → *Anwendungsname* → **Eigenschaften des Webmoduls** → **Sitzungsmanagement** → **Einstellungen für das Sitzungsmanagement**.
  2. Aktualisieren Sie die Felder, um die Sitzungspersistenz in einem Daten-Grid zu aktivieren.

  Sie können die Anwendung auch mit einem wsadmin-Script aktualisieren. Im folgenden Beispiel erstellt der Parameter **-SessionManagement** dieselbe Konfiguration, die Sie auch über die Administrationskonsole erstellen können:

  Für die Konfiguration eines fernen eXtreme-Scale-Daten-Grids:

  ```
  AdminApp.edit('DefaultApplication','[-SessionManagement[[[true
  XSRemoteSessionManagement cs0:!:grid0]]]')
  ```

  Für das integrierte eXtreme-Scale-Szenario mit Standardkonfiguration:

  ```
  AdminApp.edit('DefaultApplication','[-SessionManagement[[[true
  XSEmbeddedSessionManagement :!: :!:default]]]')
  ```

Für das integrierte eXtreme-Scale-Szenario mit einer angepassten Konfiguration:

```
AdminApp.edit('DefaultApplication','[-SessionManagement[[[true
XSEmbeddedSessionManagement :!: :!:
custom:!:c:\XS\objectgrid.xml:!:c:\XS\objectgriddeployment.xml]]]')
```

Wenn Sie die Änderungen speichern, verwendet die Anwendung das konfigurierte Daten-Grid für die Sitzungspersistenz auf dem Gerät.

- **Gehen Sie wie folgt vor, um das Sitzungsmanagement in einem vorhandenen Server zu konfigurieren:**
  1. Klicken Sie in der Administrationskonsole von WebSphere Application Server auf **Server** → **Servertypen** → **WebSphere-Anwendungsserver** → *Servername* → **Containereinstellungen** → **eXtreme Scale - Einstellungen für das Sitzungsmanagement**.
  2. Aktualisieren Sie die Felder, um die Sitzungspersistenz zu aktivieren.

  Sie können das Sitzungsmanagement auch mit den folgenden Befehlen des Tools "wsadmin" in einem vorhandenen Server konfigurieren:

  Für die Konfiguration eines fernen eXtreme-Scale-Daten-Grids:

```
AdminTask.configureServerSessionManagement('[-nodeName IBM-C77EE220EB6Node01 -serverName server1
-enableSessionManagement true -sessionManagementType XSRemoteSessionManagement -XSRemoteSessionManagement
[-catalogService cs0 -csGridName grid0]]')
```

  Für die integrierte eXtreme-Scale-Konfiguration:

  – Standardkonfiguration bei Verwendung der XML-Standarddateien:

```
AdminTask.configureServerSessionManagement('[-nodeName IBM-C77EE220EB6Node01 -serverName server1
-enableSessionManagement true -sessionManagementType XSEmbeddedSessionManagement
-XSEmbeddedSessionManagement [-embeddedGridType default -objectGridXML -objectGridDeploymentXML ]]')
```

  – Angepasste Konfiguration bei Verwendung angepasster XML-Dateien:

```
AdminTask.configureServerSessionManagement('[-nodeName IBM-C77EE220EB6Node01 -serverName server1
-enableSessionManagement true -sessionManagementType XSEmbeddedSessionManagement
-XSEmbeddedSessionManagement
[-embeddedGridType custom -objectGridXML c:\XS\objectgrid.xml -objectGridDeploymentXML
c:\XS\objectgriddeployment.xml]]')
```

  Wenn Sie die Änderungen speichern, verwendet der Server das konfigurierte Daten-Grid für die Sitzungspersistenz mit allen Anwendungen, die im Server ausgeführt werden.

- Wenn Sie weitere Aspekte der HTTP-Sitzungskonfiguration ändern möchten, können Sie die Datei `splicer.properties` bearbeiten.
  1. Klicken Sie in der Administrationskonsole von WebSphere Application Server auf **Systemverwaltung** → **Cell** → **Angepasste Eigenschaften**.
  2. Bearbeiten Sie die angepasste Eigenschaft <*app_name*>,`com.ibm.websphere.xs.sessionFilterProps`. Diese angepasste Eigenschaft gibt die Position der zu bearbeitenden Datei `splicer.properties` an. Die Datei ist im Deployment Manager enthalten.
  3. Bearbeiten Sie die Datei `splicer.properties`, die sich im Pfad für die angepasste Eigenschaft im Deployment-Manager-Profil befindet.
  4. Synchronisieren Sie Ihre Knoten so, dass die aktualisierte Datei `splicer.properties` an die Knoten weitergegeben wird. Klicken Sie auf **Systemverwaltung** → **Knoten**. Wählen Sie die Knoten aus, auf denen die Anwendung installiert ist, und klicken Sie auf **Synchronisieren**.

Ergebnisse

Sie haben den HTTP-Sitzungsmanager so konfiguriert, dass die Sitzungen in einem Daten-Grid persistent gespeichert werden.

Datei "splicer.properties":

Die Datei "splicer.properties" enthält alle Konfigurationsoptionen für einen Object-Grid-Sitzungsmanager, der auf Servlet-Filtern basiert.

Musterdatei "splicer.properties"

```
# Eigenschaftendatei, die alle Konfigurationsoptionen
# enthält, die für einen ObjectGrid-Sitzungsmanager, der auf
# Servlet-Filtern basiert, konfiguriert werden können.
#
# In dieser Eigenschaftendatei können alle Standardwerte
# eingefügt werden, die diesen Konfigurationseinstellungen zugeordnet
# werden sollen, und einzelne Einstellungen können mit
# Ant-Task-Eigenschaften überschrieben werden, wenn diese
# Eigenschaftendatei mit der ANT-Task filtersplicer verwendet wird.

# Zeichenfolgewert: "REMOTE" oder "EMBEDDED". Der Standardwert ist REMOTE.

# Damit wird angegeben, ob ein integrierter eXtreme-Scale-Container
# in jedem Anwendungsserverprozess, einschließlich WebSphere,
# WebLogic, JBoss und TomCat, gestartert werden soll.

objectGridType= REMOTE

# Ein Zeichenfolgewert, der den Namen der Grid-Instanz für eine bestimmte
# Webanwendung definiert. Der Standardname ist  sessionmanager. Wenn Sie
# diesen Parameter setzen, wird dieser Wert überschrieben.
# objectGridName =

# Es kann eine Verbindung zu einem Katalogserver aufgebaut werden, um
# eine clientseitige ObjectGrid-Instanz abzurufen. Der Wert muss das
# Format "Host:Port<,Host:Port>" haben.

# Diese Liste kann beliebig lang sein und wird nur für
# Bootstrapping verwendet.

# Die erste geeignete Adresse wird verwendet. Sie ist in WebSphere
# optional, wenn catalog.services.cluster konfiguriert ist.

catalogHostPort = Host:Port<,Host:Port>

# Ein ganzzahliger Wert, der die Zeit in Sekunden zwischen dem
# Schreiben aktualisierter Sitzungen in das ObjectGrid definiert.
# Der Standardwert sind 2 Sekunden.

replicationInterval = 1

# Ein ganzzahliger Wert, der die Anzahl der Sitzungsreferenzen definiert,
# die im Speicher gehalten werden. Der Standardwert ist 2000.

sessionTableSize =

# Zeichenfolgewert: "true" oder "false". Der Standard ist "false".
# Mit diesem Wert wird gesteuert, ob Sitzungsdaten als vollständiger
# Eintrag oder jedes Attribut gesondert gespeichert wird.

fragmentedSession = false

#  Teilt dem Sitzungsmanager mit, ob die Verwendung der URL-Codierung
# (vs. Cookies) verwendet werden soll. Der Standardwert ist "false".
```

```
useURLEncoding = false

# Eine Zeichenfolge für die Dateiposition der Datei "objectgrid.xml".
# Die integrierte XML-Datei wird automatisch geladen, wenn keine Datei
# angegeben wird oder wenn objectGridType=EMBEDDED gesetzt ist.

objectGridXML =

# Eine Zeichenfolge für die Dateiposition der XML-Datei für die
# ObjectGrid-Implementierungsrichtlinien.
# Die integrierte XML-Datei wird automatisch geladen, wenn keine Datei
# angegeben wird oder wenn objectGridType=EMBEDDED gesetzt ist.

objectGridDeploymentXML =

# Eine Zeichenfolge für die IBM WebSphere-Trace-Spezifikation.
# Auch hilfreich für alle anderen Anwendungsserver wie WebLogic.

traceSpec=

# Eine Zeichenfolge mit der Position der Trace-Datei.
# Auch hilfreich für alle anderen Anwendungsserver wie WebLogic.

traceFile=
```

WebSphere eXtreme Scale für die Verwaltung von SIP-Sitzungen verwenden:

Sie können WebSphere eXtreme Scale als Alternative zum Datenreplikationsservice (DRS) als SIP-Replikationsmechanismus (Session Initiation Protocol) für die Replikation von SIP-Sitzungen verwenden.

Konfiguration des SIP-Sitzungsmanagements

Wenn Sie WebSphere eXtreme Scale als SIP-Replikationsmechanismus verwenden möchten, definieren Sie die angepasste Eigenschaft "com.ibm.sip.ha.replicator.type". Wählen Sie in der Administrationskonsole für jeden Server, dem Sie die angepasste Eigenschaft hinzufügen möchten, **Anwendungsserver** → *mein_Anwendungsserver* → **SIP-Container** → **Angepasste Eigenschaften** aus. Geben Sie `com.ibm.sip.ha.replicator.type` im Feld "Name" und `OBJECTGRID` im Feld "Wert" ein.

Verwenden Sie die folgenden Eigenschaften, um das Verhalten des ObjectGrids anzupassen, das zum Speichern von SIP-Sitzungen verwendet wird. Klicken Sie in der Administrationskonsole für jeden Server, dem Sie die angepasste Eigenschaft hinzufügen möchten, auf **Anwendungsserver** → *mein_Anwendungsserver* → **SIP-Container** → **Angepasste Eigenschaften**. Füllen Sie die Felder **Name** und **Wert** aus. Für jeden Server müssen dieselben Eigenschaften definiert werden, damit sie ordnungsgemäß funktionieren.

Tabelle 10. Angepasste Eigenschaften für das SIP-Sitzungsmanagement mit ObjectGrid

Eigenschaft	Wert	Standardwert
com.ibm.sip.ha.replicator.type	`OBJECTGRID`: ObjectGrid als SIP-Sitzungsspeicher verwenden	
min.synchronous.replicas	Mindestanzahl synchroner Replikate	0
max.synchronous.replicas	Maximale Anzahl synchroner Replikate	0
max.asynchronous.replicas	Maximale Anzahl asynchroner Replikate	1
auto.replace.lost.shards	Weitere Informationen finden Sie im Abschnitt „Verteilte Implementierungen konfigurieren" auf Seite 167.	true

Tabelle 10. Angepasste Eigenschaften für das SIP-Sitzungsmanagement mit ObjectGrid (Forts.)

Eigenschaft	Wert	Standardwert
development.mode	• `true`: Verwendung aktiver Replikate auf demselben Knoten wie primäre Entität zulassen. • `false`: Replikate dürfen sich nicht zusammen mit der primären Entität auf demselben Knoten befinden.	false

HTTP-Sitzungsmanager für verschiedene Anwendungsserver konfigurieren

Im Lieferumfang von WebSphere eXtreme Scale ist eine Sitzungsmangementimplementierung enthalten, die den Standardsitzungsmanager für den Webcontainer überschreibt und die Sitzungsreplikation, hohe Verfügbarkeit, eine bessere Skalierbarkeit und stabile Konfigurationsoptionen bietet. Sie können den Sitzungsreplikationsmanager von eXtreme Scale und den generischen Start integrierter ObjectGrid-Container aktivieren.

Informationen zu diesem Vorgang

Sie können den HTTP-Sitzungsmanager mit anderen Anwendungsservern verwenden, in denen WebSphere Application Server nicht ausgeführt wird, z. B. WebSphere Application Server Community Edition. Zum Konfigurieren des Daten-Grids in anderen Anwendungsservern müssen Sie Ihre Anwendung verbinden und die JAR-Dateien von WebSphere eXtreme Scale in Ihre Anwendung integrieren.

Vorgehensweise

1. Fügen Sie Ihre Anwendung so zusammen, dass sie den Sitzungsmanager verwenden kann. Wenn Sie den Sitzungsmanager verwenden möchten, müssen Sie die entsprechenden Filterdeklarationen den Webimplementierungsdeskriptoren für die Anwendung hinzufügen. Außerdem werden die Konfigurationsparameter des Sitzungsmanagers in Form von Initialisierungsparametern für den Servlet-Kontext in den Implementierungsdeskriptoren an den Sitzungsmanager übergeben. Es gibt drei Methoden, mit denen Sie diese Informationen in Ihre Anwendung einführen können:
   - **Script "addObjectGridFilter"**

     Verwenden Sie ein Befehlszeilenscript, das mit eXtreme Scale bereitgestellt wird, um eine Anwendung mit Filterdeklarationen und Konfigurationen in Form von Initialisierungsparametern für den Servlet-Kontext zusammenzufügen. Das Script `objectgridRoot/session/bin/addObjectGridFilter.sh` bzw. `objectgridRoot/session/bin/addObjectGridFilter.bat` akzeptiert zwei Parameter: den absoluten Pfad der EAR-Datei der Anwendung, die Sie verbinden möchten, und den absoluten Pfad der Splicer-Eigenschaftendatei, die verschiedene Konfigurationseigenschaften enthält. Das Syntaxformat dieses Scripts ist wie folgt:

     Windows

     ```
     addObjectGridFilter.bat [Position der EAR-Datei] [Position der Eigenschaftendatei]
     ```

     UNIX

     ```
     addObjectGridFilter.sh [Position der EAR-Datei] [Position der Eigenschaftendatei]
     ```

     UNIX **Beispiel mit einer Installation von eXtreme Scale in einem eigenständigen Verzeichnis unter UNIX:**

     a. `cd ObjectGrid-Stammverzeichnis/session/bin`

b. `addObjectGridFilter.sh /tmp/mySessionTest.ear ObjectGrid-Stammverzeichnis/session/samples/splicer.properties`

Der Servlet-Filter, der eingefügt wird, verwaltet die Standardwerte für Konfigurationswerte. Sie können diese Standardwerte mit Konfigurationsoptionen überschreiben, die Sie in der Eigenschaftendatei im zweiten Argument angeben. Eine Liste der Parameter, die Sie verwenden können, finden Sie im Abschnitt „Initialisierungsparameter für den Servlet-Kontext" auf Seite 274.

Sie können die Musterdatei `splicer.properties` ändern und verwenden, die mit der Installation von eXtreme Scale bereitgestellt wird. Sie können auch das Script "addObjectGridServlets" verwenden, das den Sitzungsmanager einfügt, indem Sie jedes Servlet erweitern. Das empfohlene Script ist jedoch "addObjectGridFilter".

- **Ant-Build-Script**

  Im Lieferumfang von WebSphere eXtreme Scale ist eine Datei `build.xml` enthalten, das von Apache Ant verwendet werden kann, und im Ordner `WAS-Stammverzeichnis/bin` einer Installation von WebSphere Application Server enthalten ist. Die Datei `build.xml` kann bearbeitet werden, um die Konfigurationseigenschaften des Sitzungsmanagers zu ändern. Die Konfigurationseigenschaften sind identisch mit den Eigenschaftsnamen in der Datei `splicer.properties`. Rufen Sie nach der Änderung der Datei `build.xml` den Ant-Prozess auf, indem Sie `ant.sh`, `ws_ant.sh` (UNIX) bzw. `ant.bat`, `ws_ant.bat` (Windows) aufrufen.

- **Webdeskriptor manuell aktualisieren**

  Editieren Sie die mit der Webanwendung gepackte Datei `web.xml`, um die Filterdeklaration, die Servlet-Zuordnung und die Initialisierungsparameter für den Servlet-Kontext zu integrieren. Verwenden Sie diese Methode nicht, weil sie fehleranfällig ist.

Eine Liste der Parameter, die Sie verwenden können, finden Sie im Abschnitt „Initialisierungsparameter für den Servlet-Kontext" auf Seite 274.

2. Integrieren Sie die JAR-Dateien des Sitzungsreplikationsmanagers von WebSphere eXtreme Scale in Ihre Anwendung ein. Sie können die Dateien in das Verzeichnis `WEB-INF/lib` des Anwendungsmoduls oder in den Klassenpfad des Anwendungsservers integrieren. Die erforderlichen JAR-Dateien variieren je nach Typ der verwendeten Container:
   - Ferne eXtreme-Scale-Container: `ogclient.jar` und `sessionobjectgrid.jar`
   - Integrierte eXtreme-Scale-Container: `objectgrid.jar` und `sessionobjectgrid.jar`
3. Optional: Wenn Sie ferne eXtreme-Scale-Container verwenden, starten Sie die Container. Einzelheiten finden Sie im Abschnitt „Containerprozesse starten" auf Seite 338.
4. Implementieren Sie die Anwendung. Führen Sie dazu die Schritte aus, die Sie gewöhnlich für einen Server oder Cluster verwenden. Nach der Implementierung der Anwendung können Sie die Anwendung starten.
5. Greifen Sie auf die Anwendung zu. Sie können jetzt auf die Anwendung zugreifen, die mit dem Sitzungsmanager und mit WebSphere eXtreme Scale interagiert.

Nächste Schritte

Sie können die meisten Konfigurationsparameter des Sitzungsmanagers ändern, wenn Sie Ihre Anwendung für die Verwendung des Sitzungsmanagers instrumentieren. Zu diesen Attributen gehören Varianten des Replikationstyps (synchron

oder asynchron), die Größe der speicherinternen Sitzungstabelle usw. Abgesehen von den Attributen, die während der Instrumentierung der Anwendung geändert werden können, sind die einzigen Attribute, die Sie nach der Anwendungsimplementierung ändern können, die Attribute, die sich auf die WebSphere eXtreme Scale-Serverclustertopologie beziehen, und die Art und Weise, in der Clients (Sitzungsmanager) eine Verbindung zu diesen Servern herstellen.

Initialisierungsparameter für den Servlet-Kontext

Die folgende Liste mit Initialisierungsparametern für den Servlet-Kontext können in der Datei "splicer.properties" abhängig von der ausgewählten Splicing-Methode (Zusammenfügen) angegeben werden.

Parameter

objectGridType

Zeichenfolgewert: "REMOTE" oder "EMBEDDED". Der Standardwert ist REMOTE.

Wenn diese Einstellung auf "REMOTE" gesetzt wird, werden die Sitzungsdaten außerhalb des Servers gespeichert, auf dem die Webanwendung ausgeführt wird.

Wenn diese Einstellung auf "EMBEDDED" gesetzt wird, wird ein integrierter eXtreme-Scale-Container in dem Anwendungsserverprozess gestartet, in dem die Webanwendung ausgeführt wird.

objectGridName

Ein Zeichenfolgewert, der den Namen der ObjectGrid-Instanz definiert, die für eine bestimmte Webanwendung verwendet wird. Der Standardname ist "session".

Diese Eigenschaft muss den ObjectGrid-Namen in der ObjectGrid-XML-Datei und in der Implementierungs-XML-Datei widerspiegeln, die zum Starten der eXtreme-Scale-Container verwendet werden.

catalogHostPort

Der Katalogserver kann eine Verbindung zu einem Katalogserver aufgebaut werden, um eine clientseitige ObjectGrid-Instanz abzurufen. Der Wert muss das Format `Host:Port<,Host:Port>` haben, wobei "Host" für den Listener-Host steht, auf dem der Katalogserver ausgeführt wird, und Port für den Listener-Port dieses Katalogserverprozesses. Diese Liste kann beliebig lang sein und wird nur für Bootstrapping verwendet. Die erste geeignete Adresse wird verwendet. Sie ist in WebSphere Application Server optional, wenn die Eigenschaft "catalog.services.cluster" konfiguriert ist.

replicationInterval

Ein ganzzahliger Wert (in Sekunden), der das Intervall definiert, in dem aktualisierte Sitzungen in das ObjectGrid geschrieben werden. Der Standardwert ist 2. Die gültigen Werte sind 0 bis 60. 0 bedeutet, dass aktualisierte Sitzungen am Ende des Methodenaufrufs des Servlet-Service für jede Anforderung in das ObjectGrid geschrieben werden. Ein höherer replicationInterval-Wert verbessert die Leistung, weil weniger Aktualisierungen in das Grid geschrieben werden. Die Konfiguration ist bei einem höheren Wert jedoch weniger fehlertolerant.

Diese Einstellung gilt nur, wenn objectGridType auf "REMOTE" gesetzt ist.

sessionTableSize

Ein ganzzahliger Wert, der die Anzahl der Sitzungsreferenzen definiert, die im Speicher verwaltet werden. Der Standardwert ist 2000.

Diese Einstellung gilt nur für ferne Topologien, weil die integrierte Topologie die Sitzungsdaten bereits auf derselben Schicht wie den Webcontainer enthält.

Sitzungen werden auf der Basis der LRU-Logik aus der speicherinternen Tabelle entfernt. Wenn eine Sitzung aus der speicherinternen Tabelle entfernt wird, wird sie im Webcontainer ungültig gemacht, aber die Daten werden nicht aus dem Grid entfernt, so dass nachfolgende Anforderungen für diese Sitzungen die Daten weiterhin abrufen können.

fragmentedSession

Der Zeichenfolgewert "true" oder "false." Der Standardwert ist "true." Mit dieser Einstellung können Sie steuern, ob das Produkt Sitzungsdaten als vollständigen Eintrag oder jedes Attribut einzeln speichert.

Setzen Sie "fragmentedSession" auf "true", wenn die Webanwendungssitzung viele Attribute oder große Attribute hat. Setzen Sie "fragmentedSession" nur dann auf "false", wenn eine Sitzung wenig Attribute hat, weil alle Attribute unter demselben Schlüssel im Grid gespeichert werden.

In der früheren filterbasierten Implementierung hat diese Eigenschaft den Namen "persistenceMechanism" mit den möglichen Werten "ObjectGridStore" (fragmentiert) und "ObjectGridAtomicSessionStore" (nicht fragmentiert).

securityEnabled

Der Zeichenfolgewert "true" oder "false." Der Standardwert ist "false." Diese Einstellung aktiviert die Sicherheit für eXtreme-Scale-Clients.# Sie muss mit der Einstellung "securityEnabled" in der eXtreme-Scale-Servereigenschaftendatei übereinstimmen. Stimmen die Einstellungen nicht überein, tritt eine Ausnahme ein.

credentialGeneratorClass

Der Name der Klasse, die die Schnittstelle com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator implementiert. Diese Klasse wird verwendet, um Berechtigungsnachweise für Clients abzurufen.

credentialGeneratorProps

Die Eigenschaften für die CredentialGenerator-Implementierungsklasse. Die Eigenschaften werden mit der Methode **setProperties(String)** für das Objekt gesetzt. Der credentialGeneratorProps-Wert wird nur verwendet, wenn der Wert der Eigenschaft **credentialGeneratorClass** ungleich null ist.

objectGridXML

Die Position der Datei `objectgrid.xml`. Die integrierte XML-Datei in der eXtreme-Scale-Bibliothek wird automatisch geladen, wenn objectGridType=EMBEDDED und die Eigenschaft **objectGridXML** nicht angegeben sind.

objectGridDeploymentXML

Die Position der XML-Datei für die ObjectGrid-Implementierungsrichtlinien. Die integrierte XML-Datei in der eXtreme-Scale-Bibliothek wird automatisch geladen, wenn objectGridType=EMBEDDED und die Eigenschaft **objectGridDeploymentXML** nicht angegeben sind.

traceSpec

Die IBM WebSphere-Trace-Spezifikation als Zeichenfolgewert. Verwenden Sie diese Einstellung für andere Anwendungsserver als WebSphere Application Server.

traceFile

Die Position der Trace-Datei als Zeichenfolgewert. Verwenden Sie diese Einstellung für andere Anwendungsserver als WebSphere Application Server.

Dynamischen Cacheprovider für WebSphere eXtreme Scale konfigurieren

Die Installation und Konfiguration des dynamischen Cacheproviders für eXtreme Scale richtet sich nach Ihren Anforderungen und Ihrer konfigurierten Umgebung.

Vorbereitende Schritte

Installieren Sie den dynamischen Cacheprovider.

Zur Verwendung des dynamischen Cacheproviders muss WebSphere eXtreme Scale über die Knotenimplementierungen von WebSphere Application Server (einschließlich des Deployment-Manager-Knotens) installiert werden. Der dynamische Cacheprovider von eXtreme Scale wird in den folgenden Versionen von WebSphere Application Server unterstützt.

- WebSphere Application Server 6.1.0.25 + PK85622 und höher
- WebSphere Application Server 7.0.0.3 + PK85622 und höher

Installationsanweisungen finden Sie im Abschnitt Installation für 6.1 bzw. Installation für 7.0.

Informationen zur Verwendung des dynamischen Cacheproviders von eXtreme Scale mit IBM WebSphere Commerce finden Sie in den folgenden Artikeln in der Dokumentation zu IBM WebSphere Commerce:

- Dynamischen Cacheservice und Servlet-Caching aktivieren
- Datencache von WebSphere Commerce aktivieren

Informationen zu diesem Vorgang

Führen Sie die folgenden Schritte zum Konfigurieren des dynamischen Cacheproviders von eXtreme Scale aus:

1. Aktivieren Sie den dynamischen Cacheprovider von eXtreme Scale.

   In WebSphere Application Server 7.0 können Sie den dynamischen Cacheservice für die Verwendung des dynamischen Cacheproviders von eXtreme Scale über die Administrationskonsole oder mit einer angepassten Eigenschaft konfigurieren.

   Nach der Installation von eXtreme Scale ist der dynamische Cacheprovider von eXtreme Scale sofort als Option unter "Cacheprovider" in der Administrationskonsole verfügbar. Weitere Informationen finden Sie unter "Cacheserviceprovider auswählen".

   Sie können den dynamischen Cacheprovider von eXtreme Scale auch für eine Cacheinstanz konfigurieren, indem Sie die folgenden angepassten Eigenschaft/Wert-Paare in der Instanz festlegen. Diese angepassten Eigenschaften sind die einzige Möglichkeit, den Provider von eXtreme Scale in WebSphere Application Server 6.1 wie folgt zu aktivieren.

```
com.ibm.ws.cache.CacheConfig.cacheProviderName =
   "com.ibm.ws.objectgrid.dynacache.CacheProviderImpl"
```

Wenn Sie Ihr Caching nicht an eine spezielle definierte Objektcache- oder Servlet-Cache-Instanz weiterleiten, werden die Aufrufe der Dynamic-Cache-API wahrscheinlich von baseCache bearbeitet. BaseCache ist die Standardcacheinstanz, die als Teil des dynamischen Cacheservice erstellt wird. Dieselben Konfigurationseigenschaften werden verwendet, um die baseCache-Instanz für die Verwendung von eXtreme Scale als Cacheprovider zu konfigurieren. Diese Konfigurationseigenschaften müssen jedoch als angepasste JVM-Eigenschaften definiert werden, damit sie Auswirkungen auf baseCache haben.

Die Definition von Konfigurationsvariablen als angepasste JVM-Eigenschaften kann dazu führen, dass diese auch von anderen Caches aufgenommen werden. Objektcache- und Servlet-Cacheinstanzen sollten die in der Cacheinstanz definierten angepassten Eigenschaften bevorzugen, aber wenn Sie feststellen, dass eine bestimmte Cacheinstanz den Provider von eXtreme Scale verwendet, obwohl sie den Standardprovider verwenden sollte, kann diese Situation durch Verwendung angepasster Eigenschaften behoben werden. Damit eine Cacheinstanz den dynamischen Standardcacheprovider verwendet, definieren Sie die Eigenschaft für den Cacheprovider wie folgt:

```
com.ibm.ws.cache.CacheConfig.cacheProviderName = "default"
```

Da die für den baseCache definierten Konfigurationseigenschaften des dynamischen Cacheproviders von eXtreme Scale von anderen Cacheinstanzen verwendet werden können, sollten Sie vorsichtig sein, wenn Sie sich auf die Standardwerte für die Cachekonfigurationseigenschaften verlassen.

2. Konfigurieren Sie die Replikationseinstellung für den Cache.

   Mit dem dynamischen Cacheprovider von eXtreme Scale ist es möglich, lokale Cacheinstanzen und replizierte Cacheinstanzen zu verwenden. Bei lokalen Cacheinstanzen ist keine weitere Konfiguration erforderlich, und der Rest dieses Abschnitts kann übersprungen werden.

   Es gibt zwei Methoden für die Erstellung replizierter Caches. Die erste Methode ist die Aktivierung der Cachereplikation über die Administrationskonsole. Diese Aktivierung kann in WebSphere Application Server Version 7.0 jederzeit vorgenommen werden, setzt in WebSphere Application Server Version 6.1 aber die Erstellung einer DRS-Replikationsdomäne voraus.

   Die zweite Methode ist die Verwendung des folgenden Eigenschaft/Wert-Paars, mit dem der Cache gezwungen wird zu melden, dass er ein replizierter Cache ist, auch wenn ihm keine DRS-Replikationsdomäne zugeordnet ist.

   ```
   com.ibm.ws.cache.CacheConfig.enableCacheReplication = "true"
   ```

3. Konfigurieren Sie die Topologie für den dynamischen Cacheservice.

   Der einzige erforderliche Konfigurationsparameter für den dynamischen Cacheprovider von eXtreme Scale ist die Cachetopologie. Die folgende Variable muss als angepasste Eigenschaft im dynamischen Cacheservice definiert werden:

   ```
   com.ibm.websphere.xs.dynacache.topology
   ```

   Im Folgenden sehen Sie die drei gültigen Werte für diese Eigenschaft:

   - embedded
   - embedded_partitioned
   - remote

   Sie müssen einen der zulässigen Werte verwenden.

4. Konfigurieren Sie die Anzahl der Anfangscontainer für den dynamischen Cacheservice.

Sie können die Leistung von Caches, die die integrierte partitionierte Topologie verwenden, maximieren, indem Sie die Anzahl der Anfangscontainer konfigurieren. Die folgende Variable muss als Systemeigenschaft in der Java Virtual Machine von WebSphere Application Server definiert werden:

```
com.ibm.websphere.xs.dynacache.num_initial_containers
```

Der empfohlene Wert für diese Konfigurationseigenschaft ist eine ganze Zahl, die der Gesamtanzahl der Instanzen von WebSphere Application Server entspricht, die auf diese verteilte Cacheinstanz zugreifen, bzw. geringfügig darunter liegt. Wenn ein dynamischer Cacheservice von Grid-Membern gemeinsam genutzt wird, muss die Variable auf die Anzahl der Grid-Member gesetzt werden.

Für integrierte (embedded) und integrierte partitionierte (embedded_partitioned) Topologien müssen Sie Version 7.0 von WebSphere Application Server verwenden. Definieren Sie die folgende angepasste Eigenschaft im JVM-Prozess, um sicherzustellen, dass die Anfangscontainer sofort verfügbar sind.

```
com.ibm.ws.cache.CacheConfig.createCacheAtServerStartup=true
```

5. Konfigurieren Sie das Katalogservice-Grid von eXtreme Scale.

   Wenn Sie eXtreme Scale als dynamischen Cacheprovider für eine verteilte Cacheinstanz verwenden, müssen Sie eine Katalogservicedomäne von eXtreme Scale konfigurieren.

   Eine einzelne Katalogservicedomäne kann mehrere dynamische Cacheserviceprovider bedienen, die von eXtreme Scale gestützt werden.

   7.1+ Ein Katalogservice kann innerhalb und außerhalb von Prozessen von WebSphere Application Server ausgeführt werden. Wenn Sie ab eXtreme Scale Version 7.1 die Administrationskonsole für die Konfiguration von Domänenmechanismen für Katalogserver verwenden, verwendet der dynamische Cache diese Einstellungen. Es ist nicht erforderlich, weitere Schritte zum Konfigurieren eines Katalogservice auszuführen. Weitere Informationen finden Sie unter „Katalogservicedomänen in WebSphere Application Server erstellen” auf Seite 350. Wenn Sie eine Katalogservicedomäne ausführen, müssen Sie die angepasste Eigenschaft **catalog.services.cluster** für die Katalogserviceendpunkte definieren.

6. Konfigurieren Sie angepasste Schlüsselobjekte.

   Wenn Sie angepasste Objekte als Schlüssel verwenden, müssen die Objekte die Schnittstelle "Serializable" oder "Externalizable" implementieren. Wenn Sie die integrierte oder integrierte partitionierte Topologie verwenden, müssen Sie die Objekte in den gemeinsam genutzten Bibliothekspfad von WebSphere stellen, wie es auch der Fall ist, wenn sie mit dem dynamischen Standardcacheprovider verwendet werden. Weitere Einzelheiten finden Sie im Artikel "Schnittstellen 'DistributedMap' und 'DistributedObjectCache' für den dynamischen Cache verwenden" im Information Center von WebSphere Application Server Network Deployment.

   Wenn Sie die ferne Topologie verwenden, müssen Sie die angepassten Schlüsselobjekte für die eigenständigen Container von eXtreme Scale in den Klassenpfad stellen.

7. Konfigurieren Sie eXtreme-Scale-Containerserver.

   Die zwischengespeicherten Daten werden in eXtreme-Scale-Containern gespeichert. Container können innerhalb und außerhalb von Prozessen von WebSphere Application Server ausgeführt werden. Der Provider von eXtreme Scale erstellt automatisch Container innerhalb des WebSphere-Prozesses, wenn Sie integrierte oder integriert partitionierte Topologien für eine Cacheinstanz verwenden. Für diese Topologien ist keine weitere Konfiguration erforderlich.

Wenn Sie die ferne Topologie verwenden, müssen Sie eigenständige eXtreme-Scale-Container vor den Instanzen von WebSphere Application Server starten, die auf die Cacheinstanz zugreifen. Stellen Sie sicher, dass alle Container für einen bestimmten dynamischen Cacheservice auf dieselben Katalogserviceendpunkte zeigen.

Die XML-Konfigurationsdateien für den eigenständigen Container des dynamischen Cacheproviders von eXtreme Scale befinden sich im Verzeichnis `<Installationsstammverzeichnis>/customLibraries/ObjectGrid/dynacache/etc` für Installationen in WebSphere Application Server bzw. im Verzeichnis `<Installationsstammverzeichnis>/ObjectGrid/dynacache/etc` für eigenständige Installationen. Die Dateien haben die Namen `dynacache-remote-objectgrid.xml` und `dynacache-remote-definition.xml`. Erstellen Sie Kopien dieser Dateien, die Sie bearbeiten und verwenden, wenn Sie eigenständige Container für den dynamischen Cacheprovider von eXtreme Scale starten. Der Parameter **numIntitialContainers** in der Datei **`dynacache-remote-deployment.xml`** muss entsprechend der Anzahl der ausgeführten Containerprozesse gesetzt werden. Beachten Sie, dass das Attribut **numberOfPartitions** in der Datei `dynacache-remote-objectgrid.xml` den Standardwert 47 hat.

Anmerkung: Die Gruppe der Containerprozesse muss genügend freien Speicher haben, um alle dynamischen Cacheinstanzen zu bedienen, die für die ferne Topologie konfiguriert wurden. Alle Prozesse von WebSphere Application Server, die dieselben oder ähnliche Werte für **catalog.services.cluster** verwenden, müssen dieselbe Gruppe eigenständiger Container verwenden. Die Anzahl der Container und die Anzahl der Maschinen, auf denen sich die Container befinden, müssen entsprechend dimensioniert werden. Weitere Einzelheiten finden Sie im Abschnitt „Kapazitätsplanung und hohe Verfügbarkeit (dynamisches Caching)" auf Seite 13.

Der folgende Code enthält einen Beispiel-UNIX-Befehlszeileneintrag, der einen eigenständigen Container für den dynamischen Cacheprovider von eXtreme Scale startet:

`startOgServer.sh` *container1* `-objectGridFile` *../dynacache/etc/dynacache-remote-objectgrid.xml* `-deploymentPolicyFile` *../dynacache/etc/dynacache-remote-deployment.xml* `-catalogServiceEndpoints` *MyServer1.company.com:2809*

8. Für verteilte und integrierte Topologien aktivieren Sie den Agenten für Messungen, um die Schätzungen für die Speicherbelegung zu verbessern.

   Der Agent für Messungen schätzt die Speicherbelegung (Statistik "usedBytes"). Der Agent erfordert eine JVM der Java Version 5 oder höher.

   Laden Sie den Agenten, indem Sie der JVM-Befehlszeile das folgende Argument hinzufügen:

   `-javaagent:`*`WXS-Bibliotheksverzeichnis`*`/wxssizeagent.jar`

   Für eine integrierte Topologie fügen Sie das Argument der Befehlszeile für den Prozess von WebSphere Application Server hinzu.

   Für eine verteilte Topologie fügen Sie das Argument der Befehlszeile der Prozesse von eXtreme Scale (Container) und des Prozesses von WebSphere Application Server hinzu.

Spring-Integration konfigurieren

Spring-XML-Deskriptordatei

Sie können eine Spring-XML-Deskriptordatei verwenden, um eXtreme Scale mit Spring zu konfigurieren und zu integrieren.

In den folgenden Abschnitten finden Sie Definitionen der einzelnen Elemente und Attribute in der Spring-Datei `objectgrid.xsd`. Die Spring-Datei `objectgrid.xsd` befindet sich in der Datei `ogspring.jar` und im ObjectGrid-Namespace `com/ibm/ws/objectgrid/spring/namespace`. Ein Beispiel für das XML-Deskriptorschema finden Sie im Abschnitt „Spring-Datei `objectgrid.xsd`" auf Seite 286.

Element "register"

Verwenden Sie das Element "register", um die Standard-Bean-Factory für das ObjectGrid zu registrieren.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

id Gibt den Namen des Standard-Bean-Verzeichnisses für ein bestimmtes ObjectGrid an.

gridname
Gibt den Namen der ObjectGrid-Instanz an. Der diesem Attribut zugeordnete Wert muss mit einem gültigen ObjectGrid übereinstimmen, das in der ObjectGrid-Deskriptordatei konfiguriert ist.

```
<register
(1) id="register id"
(2) gridname="ObjectGrid-Name"
/>
```

Element "server"

Verwenden Sie das Element "server", um einen eXtreme-Scale-Server definieren, der einen Container und/oder einen Katalogservice enthalten kann.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

id Gibt den Namen des eXtreme-Scale-Servers an.

tracespec
Gibt den Trace-Typ an und aktiviert die Trace-Erstellung und die Trace-Spezifikation für den Server.

tracefile
Gibt den Pfad und den Namen der zu erstellenden und zu verwendenden Trace-Datei an.

statspec
Gibt die Statistikspezifikation für den Server an.

jmxport
Gibt die Nummer des noch nicht verwendeten Ports an, über den Sie JMX/RMI-Verbindungen aktivieren möchten. JMX unterstützt die Überwachung und Verwaltung über ferne Systeme.

isCatalog
Gibt an, ob der jeweilige Server einen Katalogservice enthält. Der Standardwert ist `false`.

name
Gibt den Namen des Servers an.

haManagerPort
Legt die Portnummer für den High Availability Manager (HA-Manager) fest.

listenerHost
Legt den Hostnamen fest, an den der ORB gebunden werden soll.

listenerPort
Legt den Port fest, an den der ORB gebunden werden soll.

maximumThreadPoolSize
Legt die maximale Anzahl der Threads im Pool fest.

memoryThresholdPercentage
Legt den Speicherschwellenwert (als Prozentsatz der maximalen Heap-Speichergröße) für die speicherbasierte Bereinigung fest.

minimumThreadPoolSize
Legt die minimale Anzahl der Threads im Pool fest.

workingDirectory
Die Eigenschaft, die das Verzeichnis definiert, das der ObjectGrid-Server für alle Standardeinstellungen verwenden soll.

zoneName
Definiert die Zone, zu der dieser Server gehört.

enableChannelFramework
Setzt die Eigenschaft "TransportMode" in den Eigenschaften des IBM ORB auf "ChannelFramework".

enableSystemStreamToFile
Definiert, ob SystemOut und SystemErr an eine Datei gesendet werden.

enableMBeans
Bestimmt, ob das ObjectGrid MBeans in diesem Prozess registriert.

serverPropertyFile
Lädt die Servereigenschaftendatei aus einer Datei.

catalogServerProperties
Gibt den Katalogserver an, der den Server enthält.

```
<server
(1) id="server id"
(2) tracespec="Trace-Spezifikation für den Server"
(3) tracefile="Trace-Datei für den Server"
(4) statspec="Statistikspezifikation für den Server"
(5) jmxport="JMX-Portnummer"
(6) isCatalog="true"|"false"
(7) name="Servername"
(8) haManagerPort="HA-Manager-Port"
(9) listenerHost="Hostname für ORB-Bindung"
(10) listenerPort="Listener-Port für ORB-Bindung"
(11) maximumThreadPoolSize="maximale Thread-Anzahl"
(12) memoryThresholdPercentage="Speicherschwellenwert (Prozentsatz der max. Heap-Speichergröße)"
(13) minimumThreadPoolSize="minimale Thread-Anzahl"
(14) workingDirectory="Position für das Arbeitsverzeichnis"
(15) zoneName="Zonenname"
(16) enableChannelFramework="true"|"false"
(17) enableSystemStreamToFile="true"|"false"
(18) enableMBeans="true"|"false"
(19 serverPropertyFile="Position der Servereigenschaftendatei"
(20) catalogServerProperties="Referenz auf Katalogservereigenschaften"
/>
```

Element "catalog"

Verwenden Sie das Element "catalog", um Containerservices im Daten-Grid weiterzuleiten.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

host
Gibt den Hostnamen der Workstation an, auf der der Service ausgeführt wird.

port
Gibt die Portnummer an, die in Kombination mit dem Hostnamen den Katalogserviceport bestimmt, zu dem der Client eine Verbindung herstellen kann.

```
<catalog
(1) host="Hostname des Katalogservice"
(2) port="Portnummer des Katalogservice"
/>
```

Element "catalog server"

Verwenden Sie das Eigenschaftselement "catalog Server", um einen Katalogserverservice zu definieren.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

catalogServerEndPoint
Gibt die Verbindungseigenschaften für den Katalogserver an.

enableQuorum
Bestimmt, ob das Quorum aktiviert wird.

heartBeatFrequencyLevel
Legt die Überwachungssignalfrequenzstufe fest.

domainName
Definiert den Domänennamen, der verwendet wird, um dieses Katalogservice-Grid für Clients eindeutig zu identifizieren, wenn Anforderungen an mehrere Domänen weitergeleitet werden.

foreignDomains
Es wird eine bidirektionale Verbindung zu jeder der fremden Domänen hergestellt, um Daten in einem Szenario mit mehreren primären Domänen auszutauschen.

clusterSecurityURL
Definiert die Position der speziellen Sicherheitsdatei für den Katalogservice.

```
<catalog server
(1) catalogServerEndPoint="Referenz auf den Katalogserverendpunkt"
(2) enableQuorum="true"|"false"
(3) heartBeatFrequencyLevel="
  HEARTBEAT_FREQUENCY_LEVEL_TYPICAL|
  HEARTBEAT_FREQUENCY_LEVEL_RELAXED|
  HEARTBEAT_FREQUENCY_LEVEL_AGGRESSIVE"
(4) domainName="Domänenname, der für die eindeutige Identifizierung des Katalogservice-Grids verwendet wird"
(5) domainEndpoints="Referenz auf die Endpunkte des Domänennamens"
(6) foreignDomains="Name der fremden Domäne"
(7) clusterSecurityURL="Position der Clustersicherheitsdatei"/>
```

Element "catalogServerEndPoint"

Verwenden Sie das Element "catalogServerEndPoint", um einen Katalogserverendpunkt für ein Element "catalog server" zu erstellen.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

serverName
Gibt den Namen an, der den zu startenden Prozess identifiziert.

hostName
Gibt den Hostnamen der Maschine an, auf der der Server gestartet wird.

clientPort
Gibt den Port an, der für die Peer-Kommunikation mit dem Katalogcluster verwendet wird.

peerPort
Gibt den Port an, der für die Peer-Kommunikation mit dem Katalogcluster verwendet wird.

```
<catalogServerEndPoint
(1) name="Name des Katalogserverendpunkts"
(2) host=""
(3) clientPort=""
(4) peerPort""
/>
```

Element "container"

Verwenden Sie das Element "container", um die Daten selbst zu speichern.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

objectgridxml
Gib den Pfad und den Namen der zu verwendenden XML-Deskriptordatei an, die die Eigenschaften für das ObjectGrid enthält, einschließlich Maps, Sperrstrategie und Plug-ins.

deploymentxml
Gibt den Pfad und den Namen der XML-Datei an, die in Kombination mit der XML-Deskriptordatei Partitionierung, Replikation, Anzahl der anfänglichen Container und weitere Einstellungen bestimmt.

server
Gibt den Server an, auf dem sich der Container befindet.

```
<server
(1) objectgridxml="ObjectGrid-XML-Deskriptordatei"
(2) deploymentxml ="ObjectGrid-XML-Implementierungsdeskriptordatei"
(3) server="Serverreferenz"
/>
```

Element "JPALoader"

Verwenden Sie das Element "JPALoader", um den ObjectGrid-Cache mit einem vorhandenen Back-End-Datenspeicher zu synchronisieren, wenn die API "ObjectMap" verwendet wird.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

entityClassName
Lässt die Verwendung von JPAs wie EntityManager.persist und EntityManager.find zu. Das Attribut **entityClassName** ist für den JPALoader erforderlich.

preloadPartition
Gibt die Partitionsnummer an, bei der der Preload-Prozess für die Map gestartet wird. Wenn der Wert kleiner als 0 oder größer als (totalNumberOfPartition – 1) ist, wird der Map-Preload-Prozess nicht gestartet.

```
<JPALoader
(1) entityClassName="Name der Entitätsklasse"
(2) preloadPartition ="int"
/>
```

Element "JPATxCallback"

Verwenden Sie das Element "JPATxCallback", um JPA- und ObjectGrid-Transaktionen zu koordinieren.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

persistenceUnitName
Erstellt eine JPA-EntityManagerFactory und sucht die Metadaten der JPA-Entität in der Datei "persistence.xml". Das Attribut **persistenceUnitName** ist erforderlich.

jpaPropertyFactory
Gibt die Factory zum Erstellen einer Map für Persistenzeigenschaften an, mit denen die Standardpersistenzeigenschaften überschrieben werden sollen. Dieses Attribut muss auf eine Bean verweisen.

exceptionMapper
Das ExceptionMapper-Plug-in kann für die Zuordnung von JPA-spezifischen Ausnahmen zu datenbankspezifischen Ausnahmen und umgekehrt verwendet werden. Dieses Attribut muss auf eine Bean verweisen.

```
<JPATxCallback
(1) persistenceUnitName="Name der JPA-Persistenzeinheit"
(2) jpaPropertyFactory ="JPAPropertyFactory-Bean-Referenz"
(3) exceptionMapper="ExceptionMapper-Bean-Referenz"
/>
```

Element "JPAEntityLoader"

Verwenden Sie das Element "JPAEntityLoader", um den ObjectGrid-Cache mit einem vorhandenen Back-End-Datenspeicher zu synchronisieren, wenn die API "EntityManager" verwendet wird.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

entityClassName
Lässt die Verwendung von JPAs wie EntityManager.persist und EntityManager-.find zu. Das Attribut **entityClassName** ist für das Element "JPAEntityLoader" optional. Wenn das Element nicht konfiguriert ist, wird die entity in der ObjectGrid-Entitäts-Map konfigurierte Entitätsklasse verwendet. Für den ObjectGrid-EntityManager und den JPA-Provider muss dieselbe Klasse verwendet werden.

preloadPartition
Gibt die Partitionsnummer an, bei der der Preload-Prozess für die Map gestartet wird. Wenn der Wert kleiner als 0 oder größer als (totalNumberOfPartition – 1) ist, wird der Map-Preload-Prozess nicht gestartet.

```
<JPAEntityLoader
(1) entityClassName="Name der Entitätsklasse"
(2) preloadPartition ="int"
/>
```

Element "LRUEvictor"

Verwenden Sie das Element "LRUEvictor", um zu entscheiden, welche Einträge entfernt werden, wenn die maximal zulässige Anzahl an Einträgen in einer Map überschritten wird.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

maxSize
Gibt die Gesamtanzahl der Einträge in einer Warteschlange an, bei der Evictor (Bereinigungsprogramm) eingreifen muss.

sleepTime
Legt das Intervall (in Sekunden) fest, in dem der Evictor die Map-Warteschlangen überprüft, um festzustellen, ob Aktionen für die Map ausgeführt werden müssen.

numberOfLRUQueues
Gibt an, wie viele Warteschlangen der Evictor durchsuchen muss, um zu verhindern, dass eine einzige Warteschlange in der Größe der gesamten Map verwendet wird.

```
<LRUEvictor
(1) maxSize="int"
(2) sleepTime ="Sekunden"
(3) numberOfLRUQueues ="int"
/>
```

Element "LFUEvictor"

Verwenden Sie das Element "LFUEvictor", um zu entscheiden, welche Einträge entfernt werden, wenn die maximal zulässige Anzahl an Einträgen in einer Map überschritten wird.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

maxSize
Gibt an, wie viele Einträge insgesamt in jedem Heap-Speicher zulässig sind, bevor der Evictor eingreifen muss.

sleepTime
Legt das Intervall (in Sekunden) fest, in dem der Evictor die Heap-Speicher der Maps überprüft, um festzustellen, ob Aktionen für die Map ausgeführt werden müssen.

numberOfHeaps
Gibt an, wie viele Heap-Speicher der Evictor durchsuchen muss, um zu verhindern, dass ein einziger Heap-Speicher in der Größe der gesamten Map verwendet wird.

```
<LFUEvictor
(1) maxSize="int"
(2) sleepTime ="Sekunden"
(3) numberOfHeaps ="int"
```

Element "HashIndex"

Verwenden Sie das Element "HashIndex" mit Java-Reflexion, um in einer Map gespeicherte Objekte zu überwachen, wenn sie aktualisiert werden.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

name
Gibt den Namen des Index an, der für jede Map eindeutig sein muss.

attributeName
Gibt den Namen des zu indexierenden Attributs an. Bei Feldzugriffsindizes entsprechen die Attributnamen den Feldnamen. Bei Eigenschaftszugriffsindizes sind die Attributnamen die JavaBean-kompatiblen Eigenschaftsnamen.

rangeIndex
Gibt an, ob die Bereichsindexierung aktiviert ist. Der Standardwert ist `false`.

fieldAccessAttribute
Wird für Maps verwendet, die keine Entitäts-Maps sind. Die Getter-Methode wird für den Zugriff auf die Daten verwendet. Der Standardwert ist `false`. Wenn diese Einstellung den Wert `true` hat, wird direkt über die Felder auf das Objekt zugegriffen.

POJOKeyIndex
Wird für Maps verwendet, die keine Entitäts-Maps sind. Der Standardwert ist `false`. Wenn Sie den Wert `true` angeben, überwacht der Index das Objekt im Schlüsselteil der Map, was hilfreich ist, wenn der Schlüssel ein zusammengesetzter Schlüssel ist und im Wert kein Schlüssel integriert ist. Wenn diese Einstellung nicht angegeben oder auf `false` gesetzt wird, überwacht der Index selbst das Objekt im Wertteil der Map.

```
<HashIndex
(1) name="Indexname"
(2) attributeName="Attributname"
(3) rangeIndex ="true"|"false"
(4) fieldAccessAttribute ="true"|"false"

(5) POJOKeyIndex ="true"|"false"
/>
```

Spring-Datei `objectgrid.xsd`

Verwenden Sie die Spring-Datei `objectgrid.xsd` für die Integration von eXtreme Scale in Spring, um eXtreme-Scale-Transaktionen zu verwalten und Clients und Server zu konfigurieren.

Beschreibungen der Elemente und Attribute, die in der Spring-Datei `objectgrid.xsd` definiert werden, finden Sie im Abschnitt „Spring-XML-Deskriptordatei” auf Seite 279.

Spring-Datei `objectgrid.xsd`

```
<xsd:schema xmlns="http://www.ibm.com/schema/objectgrid"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:beans="http://www.springframework.org/schema/beans"
     targetNamespace="http://www.ibm.com/schema/objectgrid"
    elementFormDefault="qualified"
     attributeFormDefault="unqualified">

    <xsd:import namespace="http://www.springframework.org/schema/beans" />

  <xsd:element name="transactionManager">
```

```
    <xsd:complexType>
           <xsd:attribute name="id" type="xsd:ID" />
    </xsd:complexType>
 </xsd:element>

 <xsd:element name="register">
    <xsd:complexType>
           <xsd:attribute name="id" type="xsd:ID" />
          <xsd:attribute name="gridname" type="xsd:string" />
    </xsd:complexType>
 </xsd:element>

 <xsd:element name="server">
    <xsd:complexType>
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
              <xsd:element ref="catalog" />
          </xsd:choice>
           <xsd:attribute name="id" type="xsd:ID" />
          <xsd:attribute name="tracespec" type="xsd:string" />
          <xsd:attribute name="tracefile" type="xsd:string" />
          <xsd:attribute name="statspec" type="xsd:string" />
          <xsd:attribute name="jmxport" type="xsd:integer" />
          <xsd:attribute name="isCatalog" type="xsd:boolean" />
          <xsd:attribute name="name" type="xsd:string" />
       <xsd:attribute name="haManagerPort" type="xsd:integer"/>
       <xsd:attribute name="listenerHost" type="xsd:string"/>
       <xsd:attribute name="listenerPort" type="xsd:integer"/>
       <xsd:attribute name="maximumThreadPoolSize" type="xsd:integer"/>
       <xsd:attribute name="memoryThresholdPercentage" type="xsd:integer"/>
       <xsd:attribute name="minimumThreadPoolSize" type="xsd:integer"/>
       <xsd:attribute name="workingDirectory" type="xsd:string"/>
       <xsd:attribute name="zoneName" type="xsd:string"/>
       <xsd:attribute name="enableChannelFramework" type="xsd:boolean"/>
       <xsd:attribute name="enableSystemStreamToFile" type="xsd:boolean"/>
       <xsd:attribute name="enableMBeans" type="xsd:boolean"/>
       <xsd:attribute name="serverPropertyFile" type="xsd:string"/>
       <xsd:attribute name="catalogServerProperties" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>

 <xsd:element name="catalog">
 <xsd:complexType>
   <xsd:attribute name="host" type="xsd:string" />
   <xsd:attribute name="port" type="xsd:integer" />
</xsd:complexType>
</xsd:element>

<xsd:element name="catalogServerProperties">
 <xsd:complexType>
   <xsd:choice minOccurs="0" maxOccurs="unbounded">
           <xsd:element ref="catalogServerEndPoint"/>
           </xsd:choice>
      <xsd:attribute name="id" type="xsd:ID"/>
    <xsd:attribute name="enableQuorum" type="xsd:boolean"/>
    <xsd:attribute name="heartBeatFrequencyLevel" type="xsd:integer"/>
    <xsd:attribute name="domainName" type="xsd:string"/>
    <xsd:attribute name="domainEndpoints" type="xsd:string"/>
    <xsd:attribute name="foreignDomains" type="xsd:string"/>
    <xsd:attribute name="clusterSecurityURL" type="xsd:anyURI"/>
</xsd:complexType>
</xsd:element>

<xsd:element name="catalogServerEndPoint">
  <xsd:complexType>
   <xsd:attribute name="name" type="xsd:string" />
   <xsd:attribute name="host" type="xsd:string" />
    <xsd:attribute name="clientPort" type="xsd:integer"/>
    <xsd:attribute name="peerPort" type="xsd:integer"/>
</xsd:complexType>
</xsd:element>

 <xsd:element name="container">
    <xsd:complexType>
           <xsd:attribute name="id" type="xsd:ID" />
           <xsd:attribute name="objectgridxml" type="xsd:string" />
           <xsd:attribute name="deploymentxml" type="xsd:string" />
           <xsd:attribute name="server" type="xsd:string" />
    </xsd:complexType>
  </xsd:element>

    <xsd:element name="JPALoader">
    <xsd:complexType>
           <xsd:attribute name="id" type="xsd:ID" />
          <xsd:attribute name="entityClassName" type="xsd:string" />
          <xsd:attribute name="preloadPartition" type="xsd:integer" />
    </xsd:complexType>
  </xsd:element>

   <xsd:element name="JPATxCallback">
     <xsd:complexType>
           <xsd:attribute name="id" type="xsd:ID" />
```

```
            <xsd:attribute name="persistenceUnitName" type="xsd:string" />
     <xsd:attribute name="jpaPropertyFactory" type="xsd:string" />
     <xsd:attribute name="exceptionMapper" type="xsd:string" />
 </xsd:complexType>
   </xsd:element>

    <xsd:element name="JPAEntityLoader">
      <xsd:complexType>
             <xsd:attribute name="id" type="xsd:ID" />
            <xsd:attribute name="entityClassName" type="xsd:string" />
            <xsd:attribute name="preloadPartition" type="xsd:integer" />
      </xsd:complexType>
   </xsd:element>

    <xsd:element name="LRUEvictor">
      <xsd:complexType>
             <xsd:attribute name="id" type="xsd:ID" />
            <xsd:attribute name="maxSize" type="xsd:integer" />
            <xsd:attribute name="sleepTime" type="xsd:integer" />
            <xsd:attribute name="numberOfLRUQueues" type="xsd:integer" />
            <xsd:attribute name="useMemoryUsageThresholdEviction" type="xsd:boolean" />
      </xsd:complexType>
   </xsd:element>

    <xsd:element name="LFUEvictor">
      <xsd:complexType>
             <xsd:attribute name="id" type="xsd:ID" />
            <xsd:attribute name="maxSize" type="xsd:integer" />
            <xsd:attribute name="sleepTime" type="xsd:integer" />
            <xsd:attribute name="numberOfHeaps" type="xsd:integer" />
            <xsd:attribute name="useMemoryUsageThresholdEviction" type="xsd:boolean" />
      </xsd:complexType>
   </xsd:element>

    <xsd:element name="HashIndex">
      <xsd:complexType>
             <xsd:attribute name="id" type="xsd:ID" />
            <xsd:attribute name="name" type="xsd:string" />
            <xsd:attribute name="attributeName" type="xsd:string" />
            <xsd:attribute name="rangeIndex" type="xsd:boolean" />
            <xsd:attribute name="fieldAccessAttribute" type="xsd:boolean" />
            <xsd:attribute name="POJOKeyIndex" type="xsd:boolean" />
      </xsd:complexType>
   </xsd:element>

</xsd:schema>
```

Spring-Erweiterungs-Beans und Unterstützung von Namespaces

WebSphere eXtreme Scale stellt ein Feature für die Deklaration von POJOs (Plain Old Java Object) als Erweiterungspunkte in der Datei `objectgrid.xml` und eine Methode für die Benennung der Beans und die anschließende Spezifikation des Klassennamens bereit. Normalerweise werden Instanzen der angegebenen Klasse erstellt, und diese Objekte werden dann als Plug-ins verwendet. Jetzt kann eXtreme Scale das Abrufen von Instanzen dieser Plug-in-Objekte an Spring delegieren. Wenn eine Anwendung Spring verwendet, müssen solche POJOs gewöhnlich mit dem Rest der Anwendung verbunden werden.

In manchen Fällen müssen Sie Spring für die Konfiguration bestimmter Plug-in-Objekte verwenden. Verwenden Sie die folgende Konfiguration als Beispiel:

```
<objectGrid name="Grid">
    <bean id="TransactionCallback" className="com.ibm.websphere.objectgrid.jpa.JPATxCallback">
        <property name="persistenceUnitName" type="java.lang.String"  value="employeePU" />
    </bean>
    ...
</objectGrid>
```

Die integrierte TransactionCallback-Implementierung, die Klasse "com.ibm.websphere.objectgrid.jpa.JPATxCallback", ist als TransactionCallback-Klasse konfiguriert. Diese Klasse wird, wie im vorherigen Beispiel gezeigt, mit der Eigenschaft "persistenceUnitName" konfiguriert. Die Klasse "JPATxCallback" besitzt auch das Attribut "JPAPropertyFactory", das den Typ "java.lang.Object" hat. Die ObjectGrid-XML-Konfiguration unterstützt diesen Typ von Konfiguration nicht.

Die Integration von Spring in eXtreme Scale löst dieses Problem, indem die Bean-Erstellung an das Spring-Framework delegiert wird. Die überarbeitete Konfiguration folgt:

```
<objectGrid name="Grid">
    <bean id="TransactionCallback" className="{spring}jpaTxCallback"/>
    ...
</objectGrid>
```

Die Spring-Datei für das Objekt "Grid" enthält die folgenden Informationen:

```
<bean id="jpaTxCallback" class="com.ibm.websphere.objectgrid.jpa.JPATxCallback" scope="shard">
    <property name="persistenceUnitName" value="employeeEMPU"/>
    <property name="JPAPropertyFactory" ref ="jpaPropFactory"/>
</bean>

<bean id="jpaPropFactory" class="com.ibm.ws.objectgrid.jpa.plugins.
JPAPropFactoryImpl" scope="shard">
</bean>
```

Hier ist TransactionCallback mit `{spring}jpaTxCallback` angegeben, und die Beans "jpaTxCallback" und "jpaPropFactory" werden in der Spring-Datei, wie im vorherigen Beispiel gezeigt, konfiguriert. Mit der Spring-Konfiguration kann eine JPAPropertyFactory-Bean als Parameter des JPATxCallback-Objekts konfiguriert werden.

Standard-Spring-Bean-Factory

Wenn eXtreme Scale ein Plug-in oder eine Erweiterungs-Bean (z. B. ObjectTransformer, Loader, TransactionCallback usw.) mit einem classname-Wert findet, der mit dem Präfix `{spring}` beginnt, verwendet eXtreme Scale den restlichen Teil des Namens als Namen für die Spring-Bean und ruft die Bean-Instanz über die Spring-Bean-Factory ab.

Wenn keine Bean-Factory für ein bestimmtes ObjectGrid registriert wurde, wird standardmäßig versucht, eine Datei `ObjectGridName_spring.xml` zu finden. Wenn Ihr Grid beispielsweise "Grid" heißt, hat die XML-Datei den Namen `/Grid_spring.xml`. Diese Datei muss im Klassenpfad oder in einem Verzeichnis `META-INF` im Klassenpfad enthalten sein. Wenn diese Datei gefunden wird, erstellt eXtreme Scale über diese Datei einen Anwendungskontext und über diese Bean-Factory die Beans.

Angepasste Spring-Bean-Factory

WebSphere eXtreme Scale stellt auch eine API "ObjectGridSpringFactory" bereit, über die eine Spring-Bean-Factory-Instanz für ein bestimmtes ObjectGrid registriert werden kann. Diese API registriert eine Instanz von BeanFactory mit der folgenden statischen Methode bei eXtreme Scale:

```
void registerSpringBeanFactoryAdapter(String objectGridName, Object
springBeanFactory)
```

Unterstützung von Namespaces

Seit Version 2.0 hat Spring einen Mechanismus für schemabasierte Erweiterungen für das Spring-XML-Basisformat für die Definition und Konfiguration von Beans. ObjectGrid verwendet dieses neue Feature, um ObjectGrid-Beans zu definieren und zu konfigurieren. Mit der Spring-XML-Schemaerweiterung sind einige der integrierten Implementierungen von eXtreme-Scale-Plug-ins und einige ObjectGrid-Beans im Namespace "objectgrid" vordefiniert. Wenn Sie die Spring-Konfigurations-

dateien schreiben, müssen Sie den vollständigen Klassennamen der integrierten Implementierungen nicht angeben. Stattdessen können Sie die vordefinierten Beans referenzieren.

Außerdem sinkt mit den Attributen der Beans, die im XML-Schema definiert sind, das Risiko, dass falsche Attributnamen angegeben werden. Die XML-Validierung, die auf dem XML-Schema basiert, kann diese Art von Fehlern früher im Entwicklungszyklus abfangen.

Die folgenden Beans sind in den XML-Schemaerweiterungen definiert:

- transactionManager
- register
- server
- catalog
- catalogServerProperties
- container
- JPALoader
- JPATxCallback
- JPAEntityLoader
- LRUEvictor
- LFUEvictor
- HashIndex

Diese Beans sind in der XML-Schemadatei "objectgrid.xsd" definiert. Diese XSD-Datei wird als Datei `com/ibm/ws/objectgrid/spring/namespace/objectgrid.xsd` in der Datei `ogspring.jar` geliefert. Ausführliche Beschreibungen der XSD-Datei und der in der XSD-Datei definierten Beans finden Sie in den Informationen zur Spring-Deskriptordatei im *Administratorhandbuch*.

Verwenden Sie weiterhin das JPATxCallback-Beispiel aus dem vorherigen Abschnitt. Im vorherigen Abschnitt wurde die JPATxCallback-Bean wie folgt konfiguriert:

```
<bean id="jpaTxCallback" class="com.ibm.websphere.objectgrid.jpa.JPATxCallback" scope="shard">
    <property name="persistenceUnitName" value="employeeEMPU"/>
    <property name="JPAPropertyFactory" ref ="jpaPropFactory"/>
</bean>

<bean id="jpaPropFactory" class="com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl" scope="shard">
</bean>
```

Mit dem Namespace-Feature kann die Spring-XML-Konfiguration wie folgt geschrieben werden:

```
<objectgrid:JPATxCallback id="jpaTxCallback" persistenceUnitName="employeeEMPU"
 jpaPropertyFactory="jpaPropFactory" />

<bean id="jpaPropFactory" class="com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl"
 scope="shard">
</bean>
```

Beachten Sie, dass hier die Klasse "com.ibm.websphere.objectgrid.jpa.JPATxCallback" nicht wie im vorherigen Beispiel angegeben wird, sondern dass die vordefinierte Bean "objectgrid:JPATxCallback" direkt verwendet wird. Wie Sie sehen, ist diese Konfiguration weniger ausführlich und in Bezug auf die Fehlerprüfung komfortabler.

Containerserver mit Spring-Erweiterungs-Beans starten

In diesem Beispiel wird veranschaulicht, wie Sie einen ObjectGrid-Server über Spring-verwaltete Erweiterungs-Beans und die Unterstützung von Namespaces von ObjectGrid starten.

ObjectGrid-XML-Datei

Zuerst wird eine sehr einfache ObjectGrid-XML-Datei definiert, die ein einziges ObjectGrid mit dem Namen "Grid" und eine einzige Map mit dem Namen "Test" enthält. Das ObjectGrid hat ein ObjectGridEventListener-Plug-in mit dem Namen "partitionListener" und die Map "Test" ein Evictor-Plug-in mit dem Namen "testLRUEvictor". Beachten Sie, dass sowohl das ObjectGridEventListener-Plug-in als auch das Evictor-Plug-in mit Spring konfiguriert sind, da ihre Namen "{spring}" enthalten:

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
        <objectGrid name="Grid">
     <bean id="ObjectGridEventListener" className="{spring}partitionListener" />
            <backingMap name="Test" pluginCollectionRef="test" />
        </objectGrid>
    </objectGrids>

    <backingMapPluginCollections>
        <backingMapPluginCollection id="test">
            <bean id="Evictor" className="{spring}testLRUEvictor"/>
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

ObjectGrid-XML-Implementierungsdatei

Jetzt wird eine einfache ObjectGrid-XML-Implementierungsdatei erstellt. Sie partitioniert das ObjectGrid in 5 Partitionen, und es ist kein Replikat erforderlich.

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
 xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
    <objectgridDeployment objectgridName="Grid">
        <mapSet name="mapSet" numInitialContainers="1" numberOfPartitions="5" minSyncReplicas="0"
            maxSyncReplicas="1" maxAsyncReplicas="0">
            <map ref="Test"/>
        </mapSet>
    </objectgridDeployment>
</deploymentPolicy>
```

ObjectGrid-Spring-XML-Datei

Jetzt werden die ObjectGrid-Spring-verwalteten Erweiterungs-Beans und die Unterstützung für Namespaces verwendet, um die ObjectGrid-Beans zu konfigurieren. Die Spring-XML-Datei hat den Namen "Grid_spring.xml". Beachten Sie, dass zwei Schemas in der XML-Datei enthalten sind: spring-beans-2.0.xsd ist für die Verwendung der Spring-verwalteten Beans bestimmt und objectgrid.xsd für die im Namespace "objectgrid" vordefinierten Beans:

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"
       xsi:schemaLocation="
```

```
        http://www.ibm.com/schema/objectgrid
    http://www.ibm.com/schema/objectgrid/objectgrid.xsd
        http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

     <objectgrid:register id="ogregister" gridname="Grid"/>

     <objectgrid:server id="server" isCatalog="true" name="server">
         <objectgrid:catalog host="localhost" port="2809"/>
     </objectgrid:server>

     <objectgrid:container id="container"
    objectgridxml="com/ibm/ws/objectgrid/test/springshard/objectgrid.xml"
        deploymentxml="com/ibm/ws/objectgrid/test/springshard/deployment.xml"
    server="server"/>

     <objectgrid:LRUEvictor id="testLRUEvictor" numberOfLRUQueues="31"/>

     <bean id="partitionListener"
    class="com.ibm.websphere.objectgrid.springshard.ShardListener" scope="shard"/>
</beans>
```

Es wurden 6 in dieser Spring-XML-Datei definiert:

1. *objectgrid:register*: Diese Bean registriert die Standard-Bean-Factory für das ObjectGrid "Grid".
2. *objectgrid:server*: Diese Bean definiert einen ObjectGrid-Server mit dem Namen "server". Dieser Server stellt auch Katalogservices bereit, da er eine verschachtelte Bean "objectgrid:catalog" enthält.
3. *objectgrid:catalog*: Diese Bean definiert einen ObjectGrid-Katalogserviceendpunkt, der auf "localhost:2809" gesetzt ist.
4. *objectgrid:container*: Diese Bean definiert einen ObjectGrid-Container mit der angegebenen Objectgrid-XML-Datei und der XML-Implementierungsdatei, die zuvor beschrieben wurden. Die Eigenschaft "server" gibt an, in welchem Server dieser Container ausgeführt wird.
5. *objectgrid:LRUEvictor*: Diese Bean definiert einen LRUEvictor mit der einer LRU-Warteschlangenanzahl von 31.
6. *partitionListener*: Diese Bean definiert ein ShardListener-Plug-in. Sie müssen eine Implementierung für dieses Plug-in bereitstellen. Deshalb kann sie die vordefinierten Beans nicht verwenden. Außerdem hat die Bean den Geltungsbereich "shard", d. h., es gibt nur eine einzige Instanz dieses ShardListeners pro Obejct-Grid-Shard.

Server starten

Das folgende Snippet startet den ObjectGrid-Server, in dem der Containerservice und der Katalogservice ausgeführt werden. Wie zu sehen, ist die einzige Methode, die zum Starten des Servers aufgerufen werden muss, eine Methode "get", mit der ein Bean-Container von der Bean-Factory abgerufen wird. Dies vereinfacht die Programmierungskomplexität, weil die meiste Logik in die Spring-Konfiguration verschoben wird:

```
public class ShardServer extends TestCase
{
    Container container;
    org.springframework.beans.factory.BeanFactory bf;

    public void startServer(String cep)
    {
        try
        {
            bf = new org.springframework.context.support.ClassPathXmlApplicationContext(
                "/com/ibm/ws/objectgrid/test/springshard/Grid_spring.xml", ShardServer.class);
```

```
            container = (Container)bf.getBean("container");
        }
        catch (Exception e)
        {
            throw new ObjectGridRuntimeException("Cannot start OG container", e);
        }
    }

    public void stopServer()
    {
        if(container != null)
            container.teardown();
    }
}
```

REST-Datenservice konfigurieren

Verwenden Sie die folgenden Links, um Informationen zur Verwaltung des REST-Datenservice zu suchen. Lesen Sie auch die API-Informationen zur MBean "RestService".

Eigenschaftendatei des REST-Datenservice

Zum Konfigurieren des REST-Datenservice bearbeiten Sie die Eigenschaftendatei für REST und definieren das erforderliche Entitätsschema für ein eXtreme-Scale-Grid.

Die Eigenschaftendatei des REST-Datenservice ist die Hauptkonfigurationsdatei für den REST-Datenservice von eXtreme Scale. Diese Datei ist eine typische Java Eigenschaftendatei mit Schlüssel/Wert-Paaren. Standardmäßig sucht der REST-Datenservice eine ordnungsgemäß benannte Datei `wxsRestService.properties` im Klassenpfad. Diese Datei kann auch explizit mit der folgenden Systemeigenschaft definiert werden: `wxs.restservice.props`.

```
-Dwxs.restservice.props=/usr/configs/dataservice.properties
```

Beim Laden des REST-Datenservice wird die verwendete Eigenschaftendatei in den Protokolldateien angezeigt:

```
CWOBJ4004I: Die Eigenschaftendateien für den REST-Datenservice von eXtreme
Scale REST wurden geladen: [/usr/configs/RestService.properties]
```

Die Eigenschaftendatei des REST-Datenservice unterstützt die folgenden Eigenschaften:

Tabelle 11. Eigenschaften für den REST-Datenservice

Eigenschaft	Beschreibung

catalogServiceEndPoints	Die erforderliche durch Kommas begrenzte Liste der Hosts und Ports einer Katalogservicedomäne im Format <Host:Port>. Diese Liste ist optional, wenn Sie WebSphere Application Server integriert mit eXtreme Scale für den REST-Datenservice verwenden. Einzelheiten zum Konfigurieren und Starten eines Katalogservice finden Sie in der Produktdokumentation zu WebSphere eXtreme Scale.
	
	`catalogServiceEndPoints=`
	
	`server1:2809,server2:2809`

objectGridNames	Die erforderlichen Namen der ObjectGrids, die dem REST-Service bereitgestellt werden sollen. Es ist mindestens ein ObjectGrid-Name erforderlich. Wenn Sie mehrere ObjectGrid-Namen angeben, trennen Sie diese durch Kommas: `ECommerceGrid,InventoryGrid`
objectGridClientXML	Der optionale Name der XML-Datei für das Überschreiben von ObjectGrid-Clients. Der hier angegebene Name wird aus dem Klassenpfad geladen. Die Standardeinstellung ist wie folgt: /META-INF/objectGridClient.xml. Einzelheiten zum Konfigurieren eines eXtreme-Scale-Clients finden Sie in der Produktdokumentation zu WebSphere eXtreme Scale.
objectGridNames	Die erforderlichen Namen der ObjectGrids, die dem REST-Service bereitgestellt werden sollen. Es ist mindestens ein ObjectGrid-Name erforderlich. Wenn Sie mehrere ObjectGrid-Namen angeben, trennen Sie diese durch Kommas: `ECommerceGrid,InventoryGrid`
objectGridClientXML	Der optionale Name der XML-Datei für das Überschreiben von ObjectGrid-Clients. Der hier angegebene Name wird aus dem Klassenpfad geladen. Die Standardeinstellung ist wie folgt: /META-INF/objectGridClient.xml. Einzelheiten zum Konfigurieren eines eXtreme-Scale-Clients finden Sie in der Produktdokumentation zu WebSphere eXtreme Scale.

Tabelle 11. Eigenschaften für den REST-Datenservice (Forts.)

Eigenschaft	Beschreibung
ogClientPropertyFile	Der optionale Name der ObjectGrid-Clienteigenschaftendatei. Diese Datei enthält Sicherheitseigenschaften, die für die Aktivierung der ObjectGrid-Clientsicherheit erforderlich sind. Wenn das Attribut "securityEnabled" in der Eigenschaftendatei gesetzt ist, wird die Sicherheit in dem vom REST-Service verwendeten ObjectGrid-Client aktiviert. Außerdem muss "credentialGeneratorProps" in der Eigenschaftendatei auf einen Wert mit dem Format "Benutzer:Kennwort" oder auf den Wert {xor_encoded Benutzer:Kennwort} zu setzen.
loginType	Der Typ der vom REST-Service verwendeten Authentifizierung, wenn die ObjectGrid-Clientsicherheit aktiviert ist. Wenn die ObjectGrid-Clientsicherheit nicht aktiviert ist, wird diese Eigenschaft ignoriert. Wenn die ObjectGrid-Clientsicherheit aktiviert ist und "loginType" auf "basic" gesetzt ist, verwendet der REST-Service • die in der Eigenschaft "credentialGeneratorProps" der ObjectGrid-Clienteigenschaftendatei angegebenen Berechtigungsnachweise für ObjectGrid-Operationen bei der Serviceinitialisierung. • die HTTP-Basisauthentifizierung für ObjectGrid-Sitzungsoperationen pro Anforderung. Wenn die ObjectGrid-Clientsicherheit aktiviert ist und "loginType" auf "none" gesetzt ist, verwendet der REST-Service • die in der Eigenschaft "credentialGeneratorProps" der ObjectGrid-Clienteigenschaftendatei angegebenen Berechtigungsnachweise für ObjectGrid-Operationen bei der Serviceinitialisierung. • die in der Eigenschaft "credentialGeneratorProps" der ObjectGrid-Clienteigenschaftendatei angegebenen Berechtigungsnachweise für ObjectGrid-Sitzungsoperationen pro Anforderung.
traceFile	Der optionale Name der Datei, an die die Trace-Ausgabe umgeleitet wird. Die Standardeinstellung ist "logs/trace.log".

Tabelle 11. Eigenschaften für den REST-Datenservice (Forts.)

Eigenschaft	Beschreibung
traceSpec	Die optionale Trace-Spezifikation, die der eXtreme-Scale-Laufzeitserver anfänglich verwenden soll. Die Standardeinstellung ist "*=all=disabled". Zum Durchführen eines Trace für den gesamten REST-Datenservice verwenden Sie "ObjectGridRest*=all=enabled".
verboseOutput	Wenn diese Einstellung auf "true" gesetzt ist, empfangen Clients des REST-Datenservice zusätzliche Diagnoseinformationen, wenn Fehler auftreten. Die Standardeinstellung ist "false". Diese optionale Einstellung sollte für Produktionsumgebungen auf "false" gesetzt werden, da sensible Informationen offengelegt werden können.
maxResultsPerCollection	Die optionale maximale Anzahl an Ergebnissen, die in einer Abfrage zurückgegeben werden. Der Standardwert ist "unlimited", und gültige Werte sind positive ganze Zahlen.
wxsRestAccessRightsFile	Der optionale Name der Eigenschaftendatei für die Zugriffsrechte für den REST-Datenservice von eXtreme Scale, in der die Zugriffsrechte für die Serviceoperationen und die ObjectGrid-Entitäten angegeben sind. Wenn diese Eigenschaft angegeben ist, versucht der REST-Service, die Datei aus dem angegebenen Pfad zu laden. Andernfalls versucht er, die Datei aus seinem Klassenpfad zu laden.

Konfiguration von WebSphere eXtreme Scale

Der REST-Datenservice von eXtreme Scale interagiert mit eXtreme Scale über die API "EntityManager". Ein Entitätsschema wird für ein eXtreme-Scale-Grid definiert, und die Metadaten für die Entitäten werden automatisch vom REST-Datenservice konsumiert. Einzelheiten zum Konfigurieren eines Entitätsschemas finden Sie unter Entitätsschema definieren.

Sie können beispielsweise wie folgt eine Entität definieren, die eine Person in einem eXtreme-Scale-Grid darstellt:

```
@Entity
public class Person {
  @Id String taxId;
String firstName;
  String lastName;
}
```

Tipp: Die hier verwendeten Annotationen sind im Paket `com.ibm.websphere.projector.annotations` enthalten.

Der REST-Service erstellt automatisch ein ADO.NET-EDMX-Dokument (Entity Data Model Extensions), das unter dem URI "$metadata" verfügbar ist:

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/$metadata
```

Sobald das eXtreme-Scale-Grid konfiguriert wurde und aktiv ist, muss ein eXtreme-Scale-Client konfiguriert und gepackt werden. Einzelheiten zum Konfigurieren des Clientpakets für den REST-Datenservice von eXtreme Scale finden Sie in den Informationen zum Packen und Implementieren in „REST-Datenservice installieren” auf Seite 307.

Entitätsmodell

Entitäten von WebSphere eXtreme Scale werden mithilfe der Entitätsannotationen oder einer Deskriptordatei für Entitätsmetadaten modelliert. Einzelheiten zum Konfigurieren eines eXtreme-Scale-Entitätsschemas finden Sie in den Informationen zum Definieren eines Entitätsschemas im *Programmierhandbuch*. Der REST-Datenservice von eXtreme Scale verwendet die Entitätsmetadaten für die automatische Erstellung eines EDMX-Modells für den Datenservice.

Diese Version des REST-Datenservice von WebSphere eXtreme Scale hat die folgenden Schemaeinschränkungen:

- Bei der Definition von Entitäten in einem partitionierten Grid müssen alle Entitäten eine direkte oder indirekte Einzelwertassoziation zur Stammentität haben (Schlüsselassoziation). Die Clientlaufzeitumgebung von WCF Data Services muss in der Lage sein, über die kanonische Adresse direkt auf jede Entität zugreifen zu können. Deshalb muss der Schlüssel der Stammentität, der für Partitions-Routing verwendet wird (der Schemastamm), Teil des Schlüssels in der untergeordneten Entität sein.

  Beispiel:

  ```
  @Entity(schemaRoot=true)public class Person {
      @Id String taxId;
      String firstName;
      String lastName;
      @OneToMany(mappedBy="person")
      List<Address> addresses;

  }

  @Entity
  public class Address {
      @Id int addrId;
      @Id @ManyToOne Person person;
      String street;
  }
  ```

- Es werden bidirektionale und unidirektionale Assozationen unterstützt. Unidirektionale Assoziationen funktionieren jedoch nicht immer über einen Client von Microsoft® WCF Data Services, da die Navigation nur in eine Richtung möglich ist, die Microsoft-Spezifikation aber erfordert, dass alle Assoziationen bidirektional sind.
- Referenzielle Integritätsbedingungen werden nicht unterstützt. Die Laufzeitumgebung von eXtreme Scale validiert keine Schlüssel zwischen Entitäten. Assoziationen zwischen Entitäten müssen vom Client verwaltet werden.
- Komplexe Typen werden nicht unterstützt. Die API "EntityManager" von eXtreme Scale unterstützt keine integrierbaren Attribute. Alle Attribute müssen Attribute eines einfachen Typs sein (sehen Sie sich die im Folgenden aufgelisteten einfachen Attributtypen an). Attribut, die keinen einfachen Typ haben, werden aus der Sicht des Clients als binäre Objekte behandelt.
- Entitätsvererbung wird nicht unterstützt. Die API "EntityManager" von eXtreme Scale unterstützt keine Vererbung.

- Medienressourcen und Medienlinks werden nicht unterstützt. Das Attribut "HasStream" des Elements "EntityType" im Conceptual Schema Definition Language Document for Data Services wird nicht verwendet.

Zuordnung zwischen EDM-Datentypen und Java-Datentypen

Das Protokoll "OData" definiert die folgende Liste von EDM-Typen (Entity Data Model) in seinem System abstrakter Typen. In den folgenden Abschnitten wird beschrieben, wie der REST-Adapter von eXtreme Scale den EDM-Typ auf der Basis des in der Entität definierten Basistyps auswählt. Einzelheiten zu EDM-Typen finden Sie auf der Webseite MSDN Library: Abstract Type System.

Die folgenden EDM-Typen sind in WCF Data Services verfügbar:

- Edm.Binary
- Edm.Boolean
- Edm.Byte
- Edm.DateTime
- Edm.Time
- Edm.Decimal
- Edm.Double
- Edm.Single
- Edm.Float
- Edm.Guid *
- Edm.Int16
- Edm.Int32
- Edm.Int64
- Edm.SByte
- Edm.String

Der EDM-Typ "Edm.Guid" wird vom REST-Datenservice von eXtreme Scale nicht unterstützt.

Zuordnung von Java-Typen zu EDM-Typen

Der REST-Datenservice von eXtreme Scale konvertiert Basisentitätstypen automatisch in EDM-Typen. Sie können die Typzuordnung überprüfen, indem Sie das EDMX-Metadatendokument (Entity Data Model Extensions) unter dem URI "$metadata" anzeigen. Der EDM-Typ wird von Clients für das Lesen und Schreiben von Daten aus bzw. in den REST-Datenservice verwendet.

Tabelle 12. EDM-Typen zugeordnete Java-Typen. Die Tabelle enthält die Zuordnung des für eine Entität definierten Java-Typs zum EDM-Datentyp. Beim Abrufen der Daten über eine Abfrage werden die Daten mit diesen Typen dargestellt:

Java-Typ	EDM-Typ
boolean java.lang.Boolean	Edm.Boolean
byte java.lang.Byte	Edm.SByte
short java.lang.Short	Edm.Int16
int java.lang.Integer	Edm.Int32
long java.lang.Long	Edm.Int64
float java.lang.Float	Edm.Single

Tabelle 12. EDM-Typen zugeordnete Java-Typen (Forts.). Die Tabelle enthält die Zuordnung des für eine Entität definierten Java-Typs zum EDM-Datentyp. Beim Abrufen der Daten über eine Abfrage werden die Daten mit diesen Typen dargestellt:

Java-Typ	**EDM-Typ**
double java.lang.Double	Edm.Double
java.math.BigDecimal	Edm.Decimal
java.math.BigInteger	java.math.BigInteger
java.lang.String	Edm.String
char	char
java.lang.Character	java.lang.Character
Char[]	Char[]
java.lang.Character[]	java.lang.Character[]
java.util.Calendar	Edm.DateTime
java.util.Date	java.util.Date
java.sql.Date	java.sql.Date
java.sql.Timestamp	java.sql.Timestamp
java.sql.Time	java.sql.Time
Weitere Typen	Edm.Binary

Zuordnung von EDM-Typen zu Java-Typen

Für Update- und Insert-Anforderungen geben die Nutzdaten die im REST-Datenservice von eXtreme Scale zu aktualisierenden bzw. einzufügenden Daten an. Der Service kann kompatible Datentypen automatisch in die im EDMX-Dokument definierten Datentypen konvertieren. Der REST-Datenservice konvertiert die XML-codierten Zeichenfolgedarstellungen des Werts in dem folgenden Prozess, der sich aus zwei Schritten zusammensetzt, in den richtigen Typ:

1. Es wird eine Typprüfung durchgeführt, um sicherzustellen, dass der EDM-Typ mit dem Java-Typ kompatibel ist. Ein EDM-Typ ist dann mit einem Java-Typ kompatibel, wenn die vom EDM-Typ unterstützten Daten ein Subset der Daten sind, die vom Java-Typ unterstützt werden. Der Typ "Edm.int32" ist beispielsweise mit dem Java-Typ "long" kompatibel, aber der Typ "Edm.int32" ist mit dem Java-Typ "short" nicht kompatibel.
2. Es wird ein Java-Zieltypobjekt erstellt, das den Zeichenfolgewert in den Nutzdaten darstellt.

Tabelle 13. Kompatible EDM- und Java-Typen

EDM-Typ	**Java-Typ**
Edm.Boolean	Boolean java.lang.Boolean

Tabelle 13. Kompatible EDM- und Java-Typen (Forts.)

EDM-Typ	Java-Typ
Edm.SByte	Byte java.lang.Byte short java.lang.Short int java.lang.Integer long java.lang.Long float java.lang.Float double java.lang.Double java.math.BigDecimal java.math.BigInteger char java.lang.Character
Edm.Byte, Edm.Int16	short java.lang.Short int java.lang.Integer long java.lang.Long float java.lang.Float double java.lang.Double java.math.BigDecimal java.math.BigInteger char java.lang.Character

Tabelle 13. Kompatible EDM- und Java-Typen (Forts.)

EDM-Typ	**Java-Typ**
Edm.Int32	int java.lang.Integer long java.lang.Long float java.lang.Float double java.lang.Double java.math.BigDecimal java.math.BigInteger
Edm.Int64	long java.lang.Long double java.lang.Double java.math.BigDecimal java.math.BigInteger
Edm.Double	double java.lang.Double java.math.BigDecimal
Edm.Decimal	double java.lang.Double java.math.BigDecimal java.math.BigInteger
Edm.Single	float java.lang.Float double java.lang.Double java.math.BigDecimal

Tabelle 13. Kompatible EDM- und Java-Typen (Forts.)

EDM-Typ	Java-Typ
Edm.String	java.lang.String char java.lang.Character Char[] java.lang.Character[] java.math.BigDecimal java.math.BigInteger
Edm.DateTime	java.util.Calendar java.util.Date java.sql.Date java.sql.Time java.sql.Timestamp
Edm.Time	java.sql.Time java.sql.Timestamp

Zuordnung von Zeittypen

Java enthält fünf Zeittypen für das Speichern von Datum und/oder Uhrzeit: java.util.Date, java.sql.Date, java.sql.Time, java.sql.Timestamp und java.util.Calendar. Alle Typen werden im Entitätsdatenmodell als "Edm.DateTime" ausgedrückt. Der REST-Service von eXtreme Scale konvertiert und normalisiert die Daten automatisch abhängig vom Java-Typ. In diesem Abschnitt werden verschiedene Punkte beschrieben, die Entwickler bei der Verwendung von Zeittypen beachten müssen.

Unterschiede zwischen Zeitzonen

In WCF Data Services werden die Beschreibungen von Zeitwerten im Typ "Edm.DateTime" immer mit dem UTC-Standard (Coordinated Universal Time, koordinierte Weltzeit) ausgedrückt. UTC ist der international bekannte Name für GMT (Greenwich Mean Time, mittlere Greenwich-Zeit). Die koordinierte Weltzeit ist die Zeit, die am Nullmeridian, dem UTC-Ursprung, gemessen wird. In der koordinierten Weltzeit gibt es keine Sommerzeit.

Konvertierung von Entitäts- in EDM-Typen und umgekehrt

Wenn ein Client eine Anforderung an den REST-Datenservice sendet, werden Datum und Uhrzeit als Zeit in der GMT-Zeitzone dargestellt. Beispiel:

```
"2000-02-29T21:30:30.654123456"
```

Der REST-Datenservice erstellt anschließend die entsprechende Java-Zeittypinstanz und fügt diese in die Entität im Grid ein.

Wenn ein Client eine Eigenschaft anfordert, die ein Java-Zeittyp aus dem REST-Datenservice von eXtreme Scale ist, wird der Wert immer als GMT-Zeitzonenwert normalisiert. Beispiel für eine wie folgt aufgebaute java.util.Date-Entität:

```
Calendar c = Calendar.getInstance();
c.clear();
c.set(2000, 1, 29, 21, 30, 30);
Date d = c.getTime();
```

Datum und Uhrzeit werden mit der Standardzeitzone des Java-Prozesses dargestellt, weil Calendar.getInstance() ein Calendar-Objekt mit der lokalen Zeitzone erstellt. Wenn die lokale Zeitzone CST ist, hat das vom REST-Datenservice abgerufene Datum die GMT-Zeitdarstellung: `"2000-03-01T03:30:30"`

Normalisierung von java.sql.Date

Eine eXtreme-Scale-Entität kann ein Attribut dem Java-Typ "java.sql.Date" definieren. Dieser Datentyp enthält keine Uhrzeit und wird vom REST-Datenservice normalisiert. Das bedeutet, dass die Laufzeitumgebung von eXtreme Scale keine Stunden, Minuten, Sekunden oder Millisekunden im Attribut "java.sql.Date" speichern. Ungeachtet der Zeitzonendifferenz wird das Datum immer als lokales Datum dargestellt.

Wenn der Client beispielsweise eine java.sql.Date-Eigenschaft mit dem Wert "2009-01-01T03:00:00" aktualisiert, erstellt der REST-Datenservice, der sich in der Zeitzone CST (-06:00) befindet, einfach eine java.sql.Date-Instanz, deren Uhrzeit auf "2009-01-01T00:00:00" der lokalen CST-Zeit gesetzt wird. Es wird keine Zeitzonenkonvertierung durchgeführt, um den java.sql.Date-Wert zu erstellen. Wenn der REST-Serviceclient den Wert dieses Attributs abruft, wird dieser als "2009-01-01T00:00:00Z" angezeigt. Würde eine Zeitzonenkonvertierung durchgeführt, hätte der angezeigte Wert das Datum "2008-12-31", das falsch wäre.

Normalisierung von java.sql.Time

Ähnlich wie bei java.sql.Date werden die java.sql.Time-Werte normalisiert und enthalten keine Datumsinformationen. Das bedeutet, dass die Laufzeitumgebung von eXtreme Scale Jahr, Monat und Tag nicht speichert. Die Uhrzeit wird unter Verwendung der GMT-Zeit ab dem 1. Januar 1970 gespeichert, die mit der java.sql.Time-Implementierung konsistent ist.

Wenn der Client beispielsweise eine java.sql.Time-Eigenschaft mit dem Wert "2009-01-01T03:00:00" speichert, erstellt der REST-Datenservice eine java.sql.Time-Instanz, in der die Millisekunden auf "3*60*60*1000" (entspricht 3 Stunden) gesetzt sind. Wenn der REST-Service den Wert abruft, wird dieser als "1970-01-01:03:00:00Z" angezeigt.

Assoziationen

Assoziationen definieren die Beziehung zwischen zwei Peer-Entitäten. Der REST-Service von eXtreme Scale spiegelt die Assoziationen wider, die mit Entitäten, die mit annotierten eXtreme-Scale-Entitäten definiert werden, oder mit in einer XML-Entitätsdeskriptordatei definierten Entitäten modelliert sind.

Verwaltung von Assoziationen

Der REST-Datenservice von eXtreme Scale unterstützt keine referenziellen Integritätsbedingungen. Der Client muss sicherstellen, dass Referenzen aktualisiert wer-

den, wenn Entitäten entfernt oder hinzugefügt werden. Wenn die Zielentität einer Assoziation aus dem Grid entfernt wird, aber der Link zwischen der Quellen- und der Zielentität nicht, ist der Link beschädigt. Der REST-Datenservice von eXtreme Scale und die API "EntityManager" tolerieren beschädigte Links und protokollieren diese in Form von CWPRJ1022W-Warnungen. Beschädigte Assoziationen werden einfach aus den Nutzdaten der Anforderung entfernt.

Verwenden Sie eine Stapelanforderung, um Assoziationsaktualisierungen in einer einzigen Transaktion durchzuführen, um beschädigte Links zu vermeiden. Sehen Sie sich den Abschnitt zu Stapelanforderungen an.

Das Element "ReferentialConstraint" des ADO.NET-Entitätsdatenmodells wird vom REST-Datenservice von eXtreme Scale nicht verwendet.

Assoziationsmultiplizität

Entitäten können Assoziation mit mehreren Werten oder mit einem einzelnen Wert haben. Assoziationen mit mehreren Werten oder Sammlungen sind 1:N- oder N:N-Assoziationen. Assoziationen mit einem einzelnen Wert sind 1:1- oder N:1-Assoziationen.

In einem partitionierten Grid müssen alle Entitäten einen Schlüsselassoziationspfad mit einem einzelnen Wert zu einer Stammentität haben. In einem anderen Abschnitt dieses Artikels wird beschrieben, wie eine Schlüsselassoziation definiert wird. Da die Stammentität für die Partitionierung der Entität verwendet wird, sind N:N-Assoziationen für partitionierte Grids nicht zulässig. Ein Beispiel für die Modellierung eines relationalen Entitätsschemas für ein partitioniertes Grid finden Sie unter Skalierbares Datenmodell in eXtreme Scale.

Im folgenden Beispiel wird beschrieben, wie die Assoziationstypen der API "EntityManager", die mit annotierten Java-Klassen modelliert werden, dem ADO.NET-Entitätsdatenmodell zugeordnet werden:

```
@Entity
public class Customer {
    @Id String customerId;
    @OneToOne TaxInfo taxInfo;
    @ManyToOne Address homeAddress;
    @OneToMany Collection<Order> orders;
    @ManyToMany Collection<SalesPerson> salespersons;
}

<Association Name="Customer_TaxInfo">
    <End Type="Model1.Customer" Role="Customer" Multiplicity="1" />
    <End Type="Model1.TaxInfo " Role="TaxInfo" Multiplicity="1" />
</Association>
<Association Name="Customer_Address">
    <End Type="Model1.Customer" Role="Customer" Multiplicity="1" />
    <End Type="Model1.Address" Role="TaxInfo" Multiplicity="*" />
</Association>
<Association Name="Customer_Order">
    <End Type="Model1.Customer" Role="Customer" Multiplicity="*" />
    <End Type="Model1.Order" Role="TaxInfo" Multiplicity="1" />
</Association>
<Association Name="Customer_SalesPerson">
    <End Type="Model1.Customer" Role="Customer" Multiplicity="*" />
    <End Type="Model1.SalesPerson" Role="TaxInfo" Multiplicity="*" />
</Association>
```

Bidirektionale und unidirektionale Assoziationen

Entitätsassoziationen können unidirektional oder bidirektional sein. Durch die Angabe des Attributs "mappedBy" in der Annotation @OneToOne, @OneToMany oder @ManyToMany oder des Attributs "mapped-by" im XML-Attribut-Tag "one-to-one", "one-to-many" oder "many-to-many" wird die Entität bidirektional. Das Protokoll "OData" erfordert derzeit, dass alle Entitäten bidirektional sind, was den Clients ermöglicht, Navigationspfade in beide Richtungen zu generieren. Die API "EntityManager" von eXtreme Scale ermöglicht die Modellierung unidirektionaler Assoziationen, wodurch Speicher eingespart und die Verwaltung der Assoziationen vereinfacht werden kann. Wenn eine unidirektionale Assoziation verwendet wird, dürfen die Clients des REST-Datenservice nur über die definierte Assoziation navigieren.

Beispiel: Wenn eine unidirektionale N:1-Assoziation zwischen Address und Country definiert ist, ist der folgende URI nicht zulässig.

```
/restservice/CustomerGrid/Country('USA')/addresses
```

Schlüsselassoziationen

Assoziationen mit einem Einzelwert (1:1 und N:1) können auch als vollständiger oder partieller Entitätsschlüssel eingeschlossen werden. Dies wird als Schlüsselassoziation bezeichnet.

Schlüsselassoziationen sind erforderlich, wenn ein partitioniertes Grid verwendet wird. Die Schlüsselassoziation muss für alle untergeordneten Entitäten in einem partitionierten Entitätsschema definiert werden. Das Protokoll "OData" erfordert, dass alle Entitäten direkt adressierbar sind. Das bedeutet, dass der Schlüssel in der untergeordneten Entität den für die Partitionierung verwendeten Schlüssel enthalten muss.

Im folgenden Beispiel hat Customer eine 1:N-Assoziation zu Order. Die Entität "Customer" ist die Stammentität, und das Attribut "customerId" wird für die Partitionierung der Entität verwendet. Order enthält Customer als Teil seiner Identität:

```
@Entity(schemaRoot="true")
public class Customer {
    @Id String customerId;
    @OneToMany(mappedBy="customer") Order orders
}

@Entity
public class Order {
    @Id int orderId;
    @Id @ManyToOne Customer customer;
    java.util.Date orderDate;
}
```

Wenn der REST-Datenservice das EDMX-Dokument für dieses Modell generiert, werden die Schlüsselfelder von Customer automatisch in die Entität "Order" eingeschlossen:

```
<EntityType Name="Order">
<Key>
    <PropertyRef Name="orderId"/>
    <PropertyRef Name="customer_customerId"/>
</Key>

<Property Name="orderId" Type="Edm.Int64" Nullable="false"/>
<Property Name="customer_customerId" Type="Edm.String"
    Nullable="false"/>
<Property Name="orderDate" Type="Edm.DateTime" Nullable="true"/>
```

```
<NavigationProperty Name="customer"
    Relationship="NorthwindGridModel.Customer_orders"
    FromRole="Order" ToRole="Customer"/>

<NavigationProperty Name="orderDetails"
    Relationship="NorthwindGridModel.Order_orderDetails"
    FromRole="Order" ToRole="OrderDetail"/>
</EntityType>
```

Wenn eine Entität erstellt wird, darf sich der Schlüssel nie ändern. Wenn die Schlüsselassoziation zwischen einer untergeordneten Entität und der übergeordneten Entität geändert werden muss, muss deshalb die untergeordnete Entität entfernt und mit einer anderen übergeordneten Entität neu erstellt werden. In einem partitionierten Grid erfordert dies zwei verschiedene Stapeländerungssets, da von der Verschiebung wahrscheinlich mehrere Partitionen betroffen sind.

Kaskadierende Operationen

Die API "EntityManager" unterstützt eine flexible Kaskadierungsrichtlinie. Assoziationen können für die Kaskadierung einer persist-, remove-, invalidate- oder merge-Operation markiert werden. Solche Kaskadierungsoperationen können auf einer oder beiden Seiten einer bidirektionalen Assoziation vorkommen.

Das Protokoll "OData" lässt kaskadierende delete-Operationen nur auf der 1-Seite der Assoziation zu. Die Annotation "CascadeType.REMOVE" bzw. das XML-Attribut "cascade-remove" können nicht auf beiden Seiten einer bidirektionalen 1:1-Assoziation bzw. auf der N-Seite einer 1:N-Assoziation definiert werden. Das folgende Beispiel veranschaulicht eine gültige bidirektionale Cascade.REMOVE-Assoziation:

```
@Entity(schemaRoot="true")
public class Customer {
    @Id String customerId;
    @OneToMany(mappedBy="customer", cascade=CascadeType.REMOVE)
    Order orders
}

@Entity
public class Order {
    @Id int orderId;
    @Id @ManyToOne Customer customer;
    java.util.Date orderDate;
}
```

Die generierte EDMX-Assoziation sieht folgendermaßen aus:

```
<Association Name="Customer_orders">
    <End Type="NorthwindGridModel.Customer" Role="Customer"
        Multiplicity="1">
        <OnDelete Action="Cascade"/>
    </End>
    <End Type="NorthwindGridModel.Order" Role="Order"
        Multiplicity="*"/>
</Association>
```

REST-Datenservice verwalten

Informationen zu diesem Vorgang

Verwenden Sie die folgenden Links, um Informationen zur Verwaltung des REST-Datenservice zu suchen. Sehen Sie sich auch die Informationen zur MBean "RestService" an.

REST-Datenservice installieren

In diesem Abschnitt wird beschrieben, wie der REST-Datenservice von WebSphere eXtreme Scale in einem Webserver installiert wird.

Vorbereitende Schritte

Softwarevoraussetzungen

Der REST-Datenservice von eXtreme Scale ist eine Java-Webanwendung, die in jedem Anwendungsserver implementiert werden kann, der die Java-Servlet-Spezifikation Version 2.3 und eine Java Runtime Environment der Version Version 5 oder höher unterstützt.

Die folgende Software ist erforderlich:

- Java Standard Edition 5 oder höher

  Einschränkung: Obwohl eXtreme Scale Java Standard Edition 1.4 und höher unterstützt, erfordert der REST-Datenservice Java Standard Edition 5 oder höher.
- Web-Servlet-Container Version 2.3 oder höher mit einer der folgenden Komponenten:
  - WebSphere Application Server Version 6.1.0.25 oder höher
  - WebSphere Application Server Version 7.0.0.5 oder höher
  - WebSphere Community Edition Version 2.1.1.3 oder höher
  - Apache Tomcat Version 5.5 oder höher
- eXtreme Scale Version 7.1 oder höher (einschließlich der Testversion)

Informationen zu diesem Vorgang

Der REST-Datenservice von eXtreme Scale enthält eine einzige WAR-Datei `wxsrestservice.war`. Die Datei `wxsrestservice.war` enthält ein einziges Servlet, das als Gateway zwischen Ihren WCF-Data-Services-Clientanwendungen oder einem anderen HTTP-REST-Client und einem eXtreme-Scale-Grid agiert.

Der REST-Datenservice enthält ein Muster, das Ihnen ermöglicht, schnell ein eXtreme-Scale-Grid zu erstellen und mit diesem über einen eXtreme-Scale-Client oder den REST-Datenservice zu interagieren. Einzelheiten zur Verwendung des Musters finden Sie unter "Muster und Lernprogramme zu REST-Datenservices".

Wenn eXtreme Scale 7.1 installiert oder die Testversion von eXtreme Scale Version 7.1 extrahiert wird, sind die folgenden Verzeichnisse und Dateien enthalten:

- `Ausgangsverzeichnis_des_REST-Service/lib`

  Das Verzeichnis "lib" enthält die folgenden Dateien:
  - `wxsrestservice.ear` – Der Unternehmensanwendungsarchiv des REST-Datenservice für WebSphere Application Server und WebSphere Application Server CE.
  - `wxsrestservice.war` – Das Webmodul des REST-Datenservice für Apache Tomcat.

  Die Datei `wxsrestservice.ear` enthält die Datei `wxsrestservice.war`, und beide sind eng mit der Laufzeitumgebung von WebSphere eXtreme Scale gekoppelt. Wenn ein Upgrade von eXtreme Scale auf eine neue Version vorgenommen oder

ein Fixpack angewendet wird, muss die Datei `wxsrestservice.war` bzw. `wxsrestservice.ear` manuell auf die in diesem Verzeichnis installiert Version aktualisiert werden.

- `Ausgangsverzeichnis_des_REST-Service/gettingstarted`

  Das Verzeichnis `gettingstarted` enthält ein einfaches Muster, das die Verwendung des REST-Datenservice von eXtreme Scale mit einem eXtreme-Scale-Grid veranschaulicht.

Vorgehensweise

Packen und implementieren Sie den REST-Datenservice.
Der REST-Datenservice ist als eigenständiges WAR-Modul entworfen. Zum Konfigurieren des REST-Datenservice müssen Sie die Konfiguration des REST-Datenservice und die optionalen eXtreme-Scale-Konfigurationsdateien in eine JAR-Datei oder in ein Verzeichnis packen. Dieses Anwendungspaket wird anschließend von der Laufzeitumgebung des Webcontainerservers referenziert. Die folgende Abbildung zeigt die vom REST-Datenservice von eXtreme Scale verwendeten Dateien. Die Konfigurations-JAR-Datei bzw. das Konfigurationsverzeichnis des REST-Service

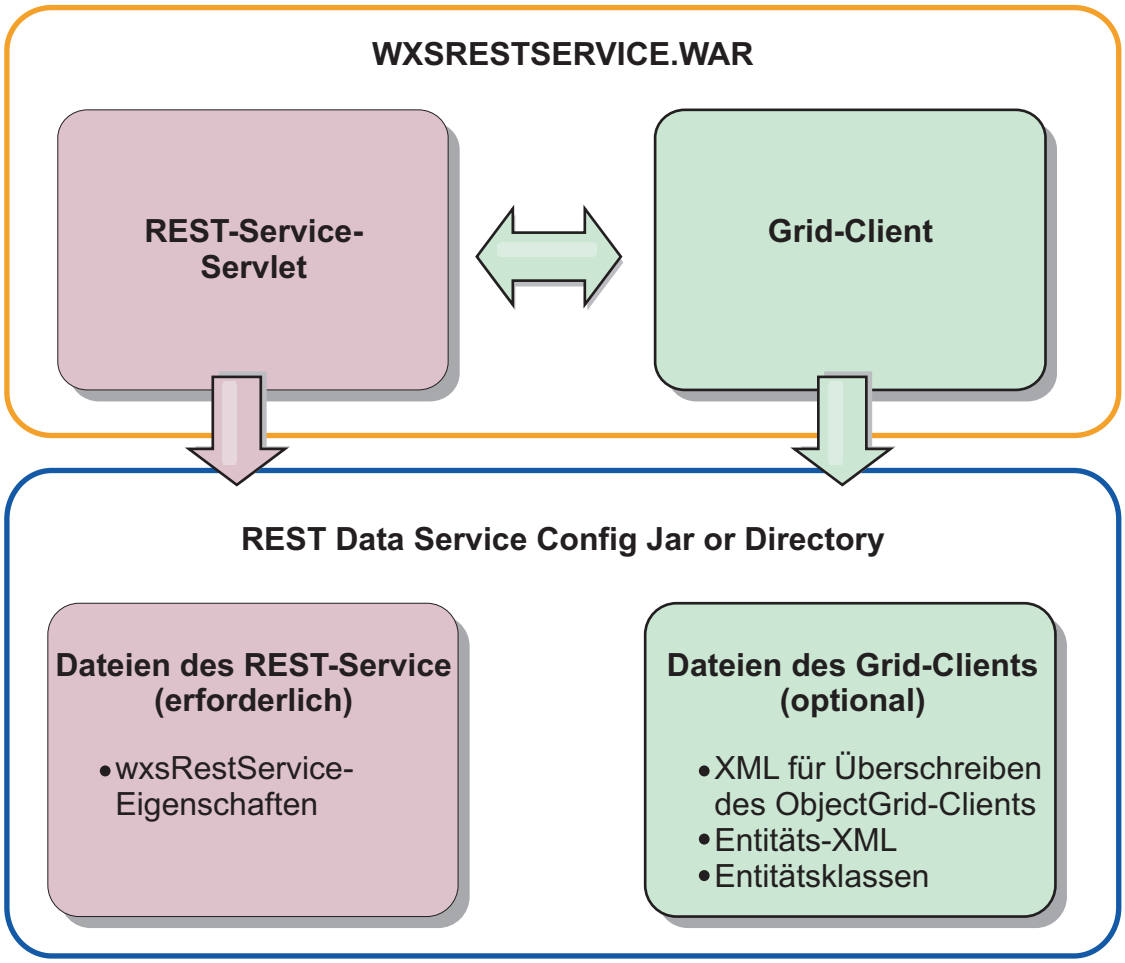


Abbildung 18. Dateien des REST-Datenservice von WebSphere eXtreme Scale

muss die folgenden Dateien enthalten:
`wxsRestService.properties`: Die Datei `wxsRestService.properties` enthält die Konfigurationsoptionen für den REST-Datenservice. Dazu gehören die Katalogserviceendpunkte, die Namen der bereitzustellenden ObjectGrids, Trace-Optionen

usw. Weitere Informationen finden Sie unter „Eigenschaftendatei des REST-Datenservice" auf Seite 293.
Die folgenden ObjectGrid-Clientdateien sind optional:

- `META-INF/objectGridClient.xml`: Die XML-Datei für das Überschreiben von ObjectGrid-Clients wird verwendet, um eine Verbindung zum fernen eXtreme-Scale-Grid herzustellen. Diese Datei ist standardmäßig nicht erforderlich. Wenn diese Datei nicht vorhanden ist, verwendet der REST-Service die Serverkonfiguration, woraufhin der nahe Cahe inaktiviert wird.

  Der Name der Datei kann mit der Konfigurationseigenschaft "objectGridClientXML" des REST-Datenservice überschrieben werden. Wenn diese XML-Datei bereitgestellt wird, muss sie Folgendes enthalten:

  1. Alle ObjectGrids, die Sie dem REST-Service bereitstellen möchten
  2. Referenz auf die XML-Entitätsdeskriptordatei, die jeder ObjectGrid-Konfiguration zugeordnet ist

- `META-INF/`*`XML-Entitätsdeskriptordateien`*: XML-Entitätsdeskriptordateien sind nur erforderlich, wenn der Client die Entitätsdefinition des Clients überschreiben muss. die XML-Entitätsdeskriptordatei muss zusammen mit der XML-Deskriptordatei für das Überschreiben von ObjectGrid-Clients verwendet werden.

  Einzelheiten zu den eXtreme-Scale-Konfigurationsdateien finden Sie im eXtreme Scale *Administratorhandbuch*.

- **Entitätsklassen**: Zum Beschreiben der Entitätsmetadaten können Sie annotierte Entitätsklassen oder eine XML-Entitätsdeskriptordatei verwenden. DER REST-Service erfordert nur dann Entitätsklassen im Klassenpfad, wenn die eXtreme-Scale-Server mit Entitätsmetadatenklassen konfiguriert wurden und keine XML-Entitätsdeskriptordatei für das Überschreiben von Clients verwendet wird.

  Im Folgenden sehen Sie ein Beispiel mit der erforderlichen Mindestkonfigurationsdatei, in dem die Entitäten in der XML auf den Servern definiert sind:

  ```
  restserviceconfig.jar:
  wxsRestService.properties
  ```

  Die Eigenschaftendatei enthält Folgendes:

  ```
  catalogServiceEndPoints=localhost:2809
  objectGridNames=NorthwindGrid
  ```

  Beispiel mit einer einzigen Entität, XML-Dateien für Überschreiben und Entitätsklassen:

  ```
  restserviceconfig.jar:
  wxsRestService.properties
  ```

  Die Eigenschaftendatei enthält Folgendes:

  ```
  catalogServiceEndPoints=localhost:2809
  objectGridNames=NorthwindGrid

  com/acme/entities/Customer.class
  META-INF/objectGridClient.xml
  ```

  Die ObjectGrid-XML-Deskriptordatei des Clients enthält Folgendes:

  ```
  <objectGrid name="CustomerGrid" entityMetadataXMLFile="emd.xml"/>
  META-INF/emd.xml
  ```

  Die XML-Deskriptordatei für die Entitätsmetadaten enthält Folgendes:

  ```
  <entity class-name="com.acme.entities.Customer" name="Customer"/>
  ```

  Einzelheiten zur API "EntityManager" und zum Konfigurieren eines eXtreme-Scale-Clients und -Servers finden Sie im *Administratorhandbuch*.

REST-Datenservice in WebSphere Application Server implementieren

In diesem Abschnitt wird beschrieben, wie Sie den REST-Datenservice von eXtreme Scale in WebSphere Application Server oder WebSphere Network Deployment Version 6.1.0.25 oder höher konfigurieren. Diese Anweisungen gelten auch für Implementierungen, in denen WebSphere eXtreme Scale mit der Implementierung von WebSphere Application Server integriert ist.

Vorbereitende Schritte

Sie müssen eine der folgenden Umgebungen auf Ihrem System haben, um den REST-Datenservice für WebSphere eXtreme Scale zu konfigurieren und zu implementieren.

- WebSphere Application Server mit dem eigenständigen eXtreme-Scale-Client:
  - WebSphere eXtreme Scale Trial Version 7.1. mit dem REST-Datenservice wurde heruntergeladen und entpackt, oder WebSphere eXtreme Scale 7.1.0.0 mit dem kumulativen Fix 2 wurde in einem eigenständigen Verzeichnis installiert.
  - WebSphere Application Server Version 6.1.0.25 oder 7.0.0.5 oder höher ist installiert und aktiv.
- WebSphere Application Server integriert mit WebSphere eXtreme Scale:

  WebSphere eXtreme Scale Version 7.1.0.0 mit dem kumulativen Fix 2 ist in WebSphere Application Server Version 6.1.0.25 oder 7.0 (oder höher) installiert.

  Tipp: Der REST-Datenservice von eXtreme Scale erfordert nur, dass die Clientoption von eXtreme Scale installiert ist. Das Profil muss nicht erweitert werden.

  Informationen zum Aktivieren der Java-2-Sicherheit finden Sie im Information Center von WebSphere Application Server.

Vorgehensweise

1. Konfigurieren und starten Sie ein eXtreme-Scale-Grid.
   a. Einzelheiten zum Konfigurieren eines eXtreme-Scale-Grids für den REST-Datenservice finden Sie in Kapitel 6, „Implementierungsumgebung konfigurieren”, auf Seite 99.
   b. Vergewissern Sie sich, dass ein eXtreme-Scale-Client eine Verbindung zum Grid herstellen und auf die Entitäten im Grid zugreifen kann. Ein Beispiel finden Sie im Abschnitt "Einführung" dieses Dokuments.
2. Erstellen Sie die Konfigurations-JAR-Datei oder das Konfigurationsverzeichnis für den REST-Datenservice von eXtreme Scale. Informationen zum Packen und Implementieren des REST-Service finden Sie unter „REST-Datenservice installieren” auf Seite 307.
3. Fügen Sie die Konfigurations-JAR-Datei bzw. das Konfigurationsverzeichnis für den REST-Datenservice dem Klassenpfad des Anwendungsservers hinzu:
   a. Öffnen Sie die WebSphere-Administrationskonsole.
   b. Navigieren Sie zu **Umgebung** → **Gemeinsam genutzte Bibliotheken**.
   c. Klicken Sie auf **Neu**.
   d. Fügen Sie den entsprechenden Feldern die folgenden Einträge hinzu:
      - Name: `extremescale_rest_configuration`
      - Klassenpfad: <Konfigurations-JAR oder -Verzeichnis für REST-Service>
   e. Klicken Sie auf **OK**.
   f. Speichern Sie die Änderungen in der Masterkonfiguration.

4. Wenn eXtreme Scale mit der Installation von WebSphere Application Server integriert ist, überspringen Sie diesen Schritt, und fahren Sie mit Schritt 5 fort. Andernfalls fahren Sie wie folgt vor:

   Fügen Sie die JAR-Datei mit der Clientlaufzeitumgebung von WebSphere eXtreme Scale (wsogclient.jar) und die Konfigurations-JAR-Datei bzw. das Konfigurationsverzeichnis für den REST-Datenservice dem Klassenpfad des Anwendungsservers hinzu:

   a. Öffnen Sie die WebSphere-Administrationskonsole.
   b. Navigieren Sie zu **Umgebung → Gemeinsam genutzte Bibliotheken**.
   c. Klicken Sie auf **Neu**.
   d. Fügen Sie den Feldern die folgenden Einträge hinzu:
      - Name: `extremescale_client_v71`
      - Klassenpfad: `WXS-Ausgangsverzeichnis/lib/wsogclient.jar`
   e. Klicken Sie auf OK.
   f. Speichern Sie die Änderungen in der Masterkonfiguration.
5. Installieren Sie die EAR-Datei des REST-Datenservice "wxsrestservice.ear" über die WebSphere-Administrationskonsole in WebSphere Application Server:
   a. Öffnen Sie die WebSphere-Administrationskonsole.
   b. Navigieren Sie zu "Anwendungen -> Neue Anwendung".
   c. Navigieren Sie zur Datei "/lib/wxsrestservice.ear" im Dateisystem, wählen Sie sie aus, und klicken Sie auf **Weiter**.
      - Wenn Sie WebSphere Application Server Version 7.0 verwenden, klicken Sie auf "Weiter".
      - Wenn Sie WebSphere Application Server Version 6.1 verwenden, geben Sie den Wert "/wxsrestservice" für den Stammkontext ein, und fahren Sie mit dem nächsten Schritt fort.
   d. Wählen Sie die Option für detaillierte Installation aus, und klicken Sie auf "Weiter".
   e. Klicken Sie in der Anzeige mit Anwendungssicherheitswarnungen auf "Weiter".
   f. Wählen Sie die Standardinstallationsoptionen aus, und klicken Sie auf "Weiter".
   g. Wählen Sie einen Server aus, dem Sie die Anwendung zuordnen möchten, und klicken Sie auf "Weiter".
   h. Verwenden Sie auf der Seite für das erneute Laden von JSP-Dateien die Standardeinstellungen, und klicken Sie auf "Weiter".
   i. Ordnen Sie auf der Seite "Gemeinsam genutzte Bibliotheken" das Modul "wxsrestservice.war" den folgenden gemeinsam genutzten Bibliotheken zu, die in den Schritten 3 und 4 definiert wurden:
      - extremescale_rest _configuration
      - extremescale_client_v71

      Tipp: Diese gemeinsam genutzte Bibliothek ist nur erforderlich, wenn eXtreme Scale nicht mit WebSphere Application Server integriert ist.
   j. Verwenden Sie auf der Seite für die Zuordnung von Beziehungen zu gemeinsam genutzten Bibliotheken die Standardeinstellungen, und klicken Sie auf "Weiter".
   k. Verwenden Sie auf der Seite für die Zuordnung virtueller Hosts die Standardeinstellungen, und klicken Sie auf "Weiter".

l. Setzen Sie auf der Seite für die Zuordnung der Kontextstammelemente das Kontextstammelement auf "wxsrestservice".

m. Klicken Sie in der Anzeige "Zusammenfassung" auf "Fertig stellen", um die Installation durchzuführen.

n. Speichern Sie die Änderungen in der Masterkonfiguration.

6. Starten Sie die REST-Datenserviceanwendung "wxsrestservice" von eXtreme Scale:

   a. Wählen Sie die Anwendung aus.

      - Wenn Sie WebSphere Application Server Version 7.0 verwenden, klicken Sie in der Administrationskonsole auf **Anwendungen** → **Anwendungstypen** → **WebSphere-Anwendungen**.
      - Wenn Sie WebSphere Application Server Version 6.1 verwenden, klicken Sie in der Administrationskonsole auf on **AnwendungenUnternehmensanwendungen**.

   b. Wählen Sie das Kontrollkästchen neben der Anwendung "wxsrestservice" aus, und klicken Sie auf **Starten**.

   c. Sehen Sie sich die Datei "SystemOut.log" für das Anwendungsserverprofil an. Wenn der REST-Datenservice erfolgreich gestartet wurde, wird die folgende Nachricht in der Datei "SystemOut.log" für das Serverprofil angezeigt:

      ```
      CWOBJ4000I: Der REST-Datenservice von WebSphere eXtreme Scale wurde gestartet.
      ```

7. Vergewissern Sie sich, dass der REST-Datenservice funktioniert: Die Portnummer finden Sie, in dem Sie in der Datei "SystemOut.log" im Verzeichnis "logs" des Anwendungsserverprofils nach dem ersten Port suchen, der für die Nachrichten-ID SRVE0250I angezeigt wird. Der Standardport ist 9080.

   Beispiel: http://localhost:9080/wxsrestservice/restservice/NorthwindGrid/. Ergebnis: Das AtomPub-Servicedokument wird angezeigt.

REST-Datenservice in WebSphere Application Server Community Edition implementieren

In diesem Abschnitt wird beschrieben, wie Sie den REST-Datenservice von eXtreme Scale in WebSphere Application Server Community Edition Version 2.1.1.3 oder höher konfigurieren.

Vorbereitende Schritte

- Eine JRE oder ein JDK von IBM (empfohlen) oder Sun der Version 5 oder höher ist installiert, und eine Umgebungsvariable JAVA_HOME ist definiert.
- Laden Sie WebSphere Application Server Community Edition Version 2.1.1.3 oder höher herunter, und installieren Sie das Produkt im `WASCE-Stammverzeichnis`, z. B. `/opt/IBM/wasce`. Informationen zu Version 2.1.1 bzw. anderen Versionen finden Sie in den Installationsanweisungen.
- eXtreme Scale Trial Version 7.1. mit dem REST-Datenservice wurde heruntergeladen und entpackt, oder WebSphere eXtreme Scale 7.1.0.0 mit dem kumulativen Fix 2 wurde in einem eigenständigen Verzeichnis installiert.

Vorgehensweise

1. Konfigurieren und starten Sie ein eXtreme-Scale-Grid.

   a. Einzelheiten zum Konfigurieren eines eXtreme-Scale-Grids für den REST-Datenservice finden Sie in Kapitel 6, „Implementierungsumgebung konfigurieren”, auf Seite 99.

b. Vergewissern Sie sich, dass ein eXtreme-Scale-Client eine Verbindung zum Grid herstellen und auf die Entitäten im Grid zugreifen kann. Ein Beispiel finden Sie unter Beispiel und Lernprogramm zum REST-Datenservice.

2. Erstellen Sie die Konfigurations-JAR-Datei oder das Konfigurationsverzeichnis für den REST-Datenservice von eXtreme Scale. Einzelheiten finden Sie in den Informationen zum Packen und Implementieren im Abschnitt „REST-Datenservice installieren" auf Seite 307.
3. Starten Sie den Server von WebSphere Application Server Community Edition:
   a. Führen Sie den folgenden Befehl aus, um den Server ohne aktivierte Java-SE-Sicherheit zu starten:

      UNIX Linux `WASCE-Stammverzeichnis/bin/startup.sh`

      Windows `WASCE-Stammverzeichnis/bin/startup.bat`

   b. Führen Sie die folgenden Schritte aus, um den Server mit aktivierter Java-SE-Sicherheit zu starten: UNIX Linux

      1) Öffnen Sie eine Befehlszeile oder ein Terminalfenster, und führen Sie den folgenden Kopierbefehl aus (oder kopieren Sie den Inhalt der angegebenen Richtliniendatei in Ihre vorhandene Richtlinie): `cp Ausgangsverzeichnis_des_REST-Service/gettingstarted/wasce/geronimo.policy WASCE-Stammverzeichnis/bin`).
      2) Bearbeiten Sie die Datei `wasce_root/bin/setenv.sh`.
      3) Fügen Sie hinter der Zeile mit `"WASCE_JAVA_HOME="` Folgendes hinzu:
      ```
      export JAVA_OPTS="-Djava.security.manager
      -Djava.security.policy=geronimo.policy".
      ```

      Windows

      1) Öffnen Sie ein Befehlszeilenfenster, und führen Sie den folgenden Kopierbefehl aus (oder kopieren Sie den Inhalt der angegebenen Richtliniendatei in Ihre vorhandene Richtlinie):
      ```
      copy Ausgangsverzeichnis_des_REST-Service\gettingstarted\wasce\
      geronimo.policy\bin
      ```
      2) Bearbeiten Sie die Datei `wasce_root\bin\setenv.bat`.
      3) Fügen Sie hinter der Zeile mit "set WASCE_JAVA_HOME=" Folgendes hinzu:
      ```
      set JAVA_OPTS="-Djava.security.manager
      -Djava.security.policy=geronimo.policy"
      ```
4. Fügen Sie die JAR-Datei für die ObjectGrid-Clientlaufzeitumgebung dem Repository von WebSphere Application Server Community Edition hinzu:
   a. Öffnen Sie die Administrationskonsole von WebSphere Application Server Community Edition, und melden Sie sich an. Der Standard-URL ist "http://localhost:8080/console". Die Standardbenutzer-ID ist "system", und das Standardkennwort ist "manager".
   b. Klicken Sie im Ordner **Services** auf der linken Seite des Konsolfensters auf den Link **Repository**.
   c. Tragen Sie im Abschnitt **Archiv dem Repository hinzufügen** Folgendes in die Eingabetextfelder ein:

Tabelle 14. Archiv dem Repository hinzufügen

Textfeld	Wert
Datei	`WXS-Ausgangsverzeichnis/lib/ogclient.jar`
Gruppe	com.ibm.websphere.xs

Tabelle 14. Archiv dem Repository hinzufügen (Forts.)

Textfeld	Wert
Artefakt	ogclient
Version	7.1
Typ	JAR

d. Klicken Sie auf die Schaltfläche "Installieren".

Suchen Sie im folgenden technischen Hinweis nach Einzelheiten zu den verschiedenen Methoden für die Konfiguration von Klassen- und Bibliotheksabhängigkeiten: Specifying external dependencies to applications running on WebSphere Application Server Community Edition.

5. Implementieren Sie das Modul des REST-Datenservice, die Datei `wxsrestservice.war`, im Server von WebSphere Application Server Community Edition, und starten Sie es.
   a. Kopieren und bearbeiten Sie die XML-Datei mit dem Musterimplementierungsplan `Ausgangsverzeichnis_des_REST-Service/gettingstarted/wasce/geronimo-web.xml`. Fügen Sie die Pfadabhängigkeiten zur Konfigurations-JAR bzw. zum Konfigurationsverzeichnis Ihres REST-Datenservice hinzu. Sehen Sie sich das Beispiel zum Definieren des Klassenpfads an, um Ihre Datei `wxsRestService.properties` sowie andere Konfigurationsdateien und Metadatenklassen hinzuzufügen.
   b. Öffnen Sie die Administrationskonsole von WebSphere Application Server Community Edition, und melden Sie sich an.

      Tipp: Der Standard-URL ist "http://localhost:8080/console". Die Standardbenutzer-ID ist "system", und das Standardkennwort ist "manager".
   c. Klicken Sie auf den Link **Neu Implementieren** auf der linken Seite des Konsolfensters.
   d. Geben Sie auf der Seite **Neue Anwendungen installieren** die folgenden Werte in die Textfelder ein:

Tabelle 15. Neue Anwendungen installieren

Textfeld	Wert
Archiv	Ausgangsverzeichnis_des_REST-Service/lib/wxsrestservice.war
Plan	Ausgangsverzeichnis_des_REST-Service/gettingstarted/wasce/geronimo-web.xml

      Tipp: Verwenden Sie den Pfad zur Datei `geronimo-web.xml`, die Sie in Schritt 3 kopiert und bearbeitet haben.
   e. Klicken Sie auf die Schaltfläche "Installieren". Auf der Konsolseite sollte daraufhin angezeigt werden, dass die Anwendung erfolgreich installiert und gestartet werden.
   f. Überprüfen Sie anhand des Systemausgabeprotokolls von WebSphere Application Server Community Edition oder der Konsole, ob der REST-Datenservice erfolgreich gestartet wurde, indem Sie nach der folgenden Nachricht suchen:

      ```
      CWOBJ4000I: Der REST-Datenservice von WebSphere eXtreme Scale wurde gestartet.
      ```
6. Starten Sie den Server von WebSphere Application Server Community Edition mit dem folgenden Befehl:

- UNIX Linux `WASCE-Stammverzeichnis/bin/startup.sh`
- Windows `WASCE-Stammverzeichnis/bin/startup.bat`

7. Installieren Sie den REST-Datenservice von eXtreme Scale und das bereitgestellte Muster im Server von WebSphere Application Server Community Edition:
   a. Fügen Sie die JAR-Datei für die ObjectGrid-Clientlaufzeitumgebung dem Repository von WebSphere Application Server Community Edition hinzu:
      1) Öffnen Sie die Administrationskonsole von WebSphere Application Server Community Edition, und melden Sie sich an. (Die Standardeinstellung ist "http://localhost:8080/console/" mit der Benutzer-ID "system" und dem Kennwort "manager".)
      2) Klicken Sie im Ordner "Services" auf der linken Seite des Konsolfensters auf den Link "**Repository**".
      3) Tragen Sie im Abschnitt **Archiv dem Repository hinzufügen** Folgendes in die Eingabetextfelder ein:

Tabelle 16. Archiv dem Repository hinzufügen

Textfeld	**Wert**
Datei	WXS-Ausgangsverzeichnis/lib/ogclient.jar
Gruppe	com.ibm.websphere.xs
Artefakt	ogclient
Version	7.1
Typ	JAR

      4) Klicken Sie auf die Schaltfläche "Installieren".

         Tipp: Suchen Sie im folgenden technischen Hinweis nach Einzelheiten zu den verschiedenen Methoden für die Konfiguration von Klassen- und Bibliotheksabhängigkeiten: Specifying external dependencies to applications running on WebSphere Application Server Community Edition.
   b. Implementieren Sie das REST-Datenservicemodul `wxsrestservice.war` im Server von WebSphere Application Server Community Edition.
      1) Bearbeiten Sie die XML-Musterimplementierungsdatei `Ausgangsverzeichnis_des_REST-Service/gettingstarted/wasce/geronimo-web.xml`, und fügen Sie den Klassenpfadverzeichnissen für das Einführungsmuster Pfadabhängigkeiten hinzu.
         - Ändern Sie die classesDirs-Pfade für die beiden GBeans des Getting-Started-Clients:

         Der classesDirs-Pfad für die GBean "GettingStarted_Client_SharedLib" muss auf `Ausgangsverzeichnis_des_REST-Service/Ggettingstarted/restclient/bin` gesetzt werden.

         Der classesDirs-Pfad für die GBean "GettingStarted_Common_Shared-Lib" muss auf `Ausgangsverzeichnis_des_REST-Service/gettingstarted/common/bin` gesetzt werden.
      2) Öffnen Sie die Administrationskonsole von WebSphere Application Server Community Edition, und melden Sie sich an.
      3) Klicken Sie auf den Link **Neu Implementieren** auf der linken Seite des Konsolfensters.
      4) Geben Sie auf der Seite **Neue Anwendungen installieren** die folgenden Werte in die Textfelder ein:

Tabelle 17. Neue Anwendungen installieren

Textfeld	Wert
Archiv	*Ausgangsverzeichnis_des_REST-Service*/lib/wxsrestservice.war
Plan	*Ausgangsverzeichnis_des_REST-Service*/gettingstarted/wasce/geronimo-web.xml

5) Klicken Sie auf die Schaltfläche **Installieren**.

   Auf der Konsolseite sollte daraufhin angezeigt werden, dass die Anwendung erfolgreich installiert und gestartet werden.

6) Überprüfen Sie anhand des Systemausgabeprotokolls von WebSphere Application Server Community Edition, ob der REST-Datenservice erfolgreich gestartet wurde, indem Sie nach der folgenden Nachricht suchen:

   ```
   CWOBJ4000I: Der REST-Datenservice von WebSphere eXtreme Scale
   wurde gestartet.
   ```

8. Vergewissern Sie sich, dass der REST-Datenservice funktioniert:

   Öffnen Sie einen Webbrowser, und navigieren Sie zum folgenden URL:
   `http://<Host>:<Port>/<Kontextstammelement >/restservice/<Grid-Name>`

   Der Standardport für WebSphere Application Server Community Edition ist 8080 und wird mit der Eigenschaft "HTTPPort" in der Datei `/var/config/config-substitutions.properties` definiert.

   Beispiel:http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/

Ergebnisse

Das Dokument des AtomPub-Service wird angezeigt.

REST-Datenservice in Apache Tomcat implementieren

In diesem Abschnitt wird beschrieben, wie Sie den REST-Datenservice von WebSphere eXtreme Scale in Apache Tomcat Version 5.5 und höher konfigurieren.

Informationen zu diesem Vorgang

- Eine JRE oder ein JDK von IBM oder Sun der Version 5 oder höher ist installiert, und eine Umgebungsvariable JAVA_HOME ist definiert.
- Apache Tomcat Version 5.5 oder höher ist installiert. Einzelheiten zur installation von Tomcat finden Sie auf der Webseite Apache Tomcat.
- eXtreme Scale Trial Version 7. mit dem REST-Datenservice wurde heruntergeladen und entpackt, oder WebSphere eXtreme Scale Version 7.1.0.0 mit dem kumulativen Fix 2 wurde in einem eigenständigen Verzeichnis installiert.

Vorgehensweise

1. Wenn Sie eine Sun JRE oder ein Sun JDK verwenden, installieren Sie IBM ORB in Tomcat:
   a. Tomcat Version 5.5:

      Kopieren Sie alle JAR-Dateien aus dem

      Verzeichnis *WXS-Ausgangsverzeichnis*/lib/endorsed

      in das

      Verzeichnis *Tomcat-Stammverzeichnis*/common/endorsed.

   b. Tomcat Version 6.0:

      Erstellen Sie ein Verzeichnis "endorsed":

UNIX Linux `mkdir Tomcat-Stammverzeichnis/endorsed`

Windows `md Tomcat-Stammverzeichnis/endorsed`

Kopieren Sie alle JAR-Dateien aus dem Verzeichnis `wxs_home/lib/endorsed` in das Verzeichnis `tomcat_root/common/endorsed`.

2. Konfigurieren und starten Sie ein eXtreme-Scale-Grid.
   a. Einzelheiten zum Konfigurieren eines eXtreme-Scale-Grids für den REST-Datenservice finden Sie in Kapitel 6, „Implementierungsumgebung konfigurieren”, auf Seite 99.
   b. Vergewissern Sie sich, dass ein eXtreme-Scale-Client eine Verbindung zum Grid herstellen und auf die Entitäten im Grid zugreifen kann. Ein Beispiel finden Sie unter Beispiel und Lernprogramm zum REST-Datenservice.
3. Erstellen Sie die Konfigurations-JAR-Datei oder das Konfigurationsverzeichnis für den REST-Datenservice von eXtreme Scale. Einzelheiten finden Sie in den Informationen zum Packen und Implementieren unter „REST-Datenservice installieren” auf Seite 307.
4. Implementieren Sie das REST-Datenservicemodul "wxsrestservice.war" im Tomcat-Server.

   Kopieren Sie die Datei "wxsrestservice.war" von `Ausgangsverzeichnis_des_REST-Service/lib` in das `Tomcat-Stammverzeichnis/webapps`
5. Fügen Sie die JAR-Datei für die ObjectGrid-Clientlaufzeitumgebung und die Anwendungs-JAR-Datei dem gemeinsam genutzten Klassenpfad in Tomcat hinzu:
   a. Bearbeiten Sie die Datei `Tomcat-Stammverzeichnis/conf/catalina.properties`.
   b. Fügen Sie die folgenden Pfadnamen am Ende der Eigenschaft "shared.loader" hinzu, indem Sie die einzelnen Pfadnamen durch Kommas trennen:
      - `WXS-Ausgangsverzeichnis/lib/ogclient.jar`
      - `Ausgangsverzeichnis_des_REST-Service/gettingstarted/restclient/bin`
      - `Ausgangsverzeichnis_des_REST-Service//gettingstarted/common/bin`
6. Wenn Sie die Java-2-Sicherheit verwenden, fügen Sie der Tomcat-Richtliniendatei Sicherheitsberechtigungen hinzu:
   - Bei der Verwendung von Tomcat Version 5.5:

     Führen Sie den Inhalt der Musterdatei `catalina policy` der Version 5.5 in `Ausgangsverzeichnis_des_REST-Service/gettingstarted/tomcat/catalina-5_5.policy` und den Inhalt der Datei `Tomcat-Stammverzeichnis/conf/catalina.policy` zusammen.
   - Bei der Verwendung von Tomcat Version 6.0:

     Führen Sie den Inhalt der Musterdatei `catalina policy` der Version 6.0 in `Ausgangsverzeichnis_des_REST-Service/gettingstarted/tomcat/catalina-6_0.policy` mit dem Inhalt der Datei `Tomcat-Stammverzeichnis/conf/catalina.policy` zusammen.
7. Starten Sie den Tomcat-Server:
   - **Bei der Verwendung von Tomcat 5.5 unter UNIX oder Windows oder der Verwendung von Tomcat 6.0 durch Verteilung über eine ZIP-Datei**

a. `cd Tomcat-Stammverzeichnis/bin`
b. Starten Sie den Server:
   - Ohne aktivierte Java-2-Sicherheit:

     UNIX Linux `./catalina.sh run`

     Windows `catalina.bat run`
   - Mit aktivierter Java-2-Sicherheit:

     UNIX Linux `./catalina.sh run -security`

     Windows `catalina.bat run -security`
c. Die Apache-Tomcat-Protokolle werden in der Konsole angezeigt. Wenn der REST-Datenservice erfolgreich gestartet wurde, wird die folgende Nachricht in der Administrationskonsole angezeigt:

   `CWOBJ4000I: Der REST-Datenservice von WebSphere eXtreme Scale wurde gestartet.`

- **Bei der Verwendung von Tomcat 6.0 unter Windows über Verteilung durch das Windows-Installationsprogramm:**
  a. cd /bin
  b. Starten Sie das Konfigurationstool von Apache Tomcat 6:

     `tomcat6w.exe`
  c. Java-2-Sicherheit aktivieren (optional):

     Fügen Sie den Java-Optionen auf der Registerkarte "Java" im Eigenschaftsfenster von Apache Tomcat 6 die folgenden Einträge hinzu:

     `-Djava.security.manager`

     `-Djava.security.policy=\conf\catalina.policy`
  d. Klicken Sie im Eigenschaftsfenster von Apache Tomcat 6 auf die Startschaltfläche, um den Tomcat-Server zu starten.
  e. Sehen Sie sich die folgenden Protokolle an, um sich zu vergewissern, dass der Tomcat-Server erfolgreich gestartet wurde:
     - `Tomcat-Stammverzeichnis/bin/catalina.log`

       Zeigt den Status der Tomcat-Server-Engine an.
     - `Tomcat-Stammverzeichnis/bin/stdout.log`

       Zeigt das Systemausgabeprotokoll an.
  f. Wenn der REST-Datenservice erfolgreich gestartet wurde, wird die folgende Nachricht im Systemausgabeprotokoll angezeigt:

     `CWOBJ4000I: Der REST-Datenservice von WebSphere eXtreme Scale wurde gestartet.`

8. Vergewissern Sie sich, dass der REST-Datenservice funktioniert.

   Öffnen Sie einen Webbrowser, und navigieren Sie zum folgenden URL:

   `http://`*`Host:Port`*`/`*`Kontextstammelement`*`/restservice/`*`Grid-Name`*

   Der Standardport für Tomcat ist 8080 und wird in der Datei `Tomcat-Stammverzeichnis/conf/server.xml` im Element <Connector> definiert.

   Beispiel:

   `http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/`

Ergebnisse

Das Dokument des AtomPub-Service wird angezeigt.

REST-Datenservice sichern

Sie können diverse Aspekte des REST-Datenservice sichern. Der Zugriff auf den REST-Datenservice von eXtreme Scale kann mit Authentifizierung und Berechtigung gesichert werden. Außerdem kann der Zugriff durch servicebezogene Konfigurationsregel, so genannte Zugriffsregeln, gesteuert werden. Die Transportsicherheit ist der dritte Aspekt.

Informationen zu diesem Vorgang

Der Zugriff auf den REST-Datenservice von eXtreme Scale kann mit Authentifizierung und Berechtigung gesichert werden. Authentifizierung und Berechtigung werden durch die Integration mit der Sicherheit von eXtreme Scale erreicht.

Der Zugriff kann auch über servicebezogene Konfigurationsregeln, so genannte Zugriffsregeln, gesteuert werden. Es gibt zwei Typen von Zugriffsregeln: Serviceoperationsrechte, die die CRUD-Operationen steuern, die der Service zulässt, und Entitätszugriffsrechte, die die CRUD-Operationen steuern, die für einen bestimmten Entitätstyp zulässig sind.

Die Transportsicherheit wird über die Hostcontainerkonfiguration (für Verbindungen vom Webclient zum REST-Service) und über die Clientkonfiguration von eXtreme Scale (für Verbindungen vom REST-Service zum eXtreme-Scale-Grid) bereitgestellt.

Vorgehensweise

- Steuerung der Authentifizierung und der Berechtigung.

  Der Zugriff auf den REST-Datenservice von eXtreme Scale kann mit Authentifizierung und Berechtigung gesichert werden. Authentifizierung und Berechtigung werden durch die Integration mit der Sicherheit von eXtreme Scale erreicht.

  Der REST-Datenservice von eXtreme Scale verwendet die Sicherheit von eXtreme Scale (Authentifizierung und Berechtigung), um zu steuern, welche Benutzer auf den Service zugreifen, und welche Operationen ein Benutzer über den Service ausführen darf. Der REST-Datenservice von eXtreme Scale verwendet entweder einen konfigurierten globalen Berechtigungsnachweis (Benutzer und Kennwort) oder einen Berechtigungsnachweis, der aus einer HTTP-Basisanforderung abgeleitet wird, die mit jeder Transaktion an das eXtreme-Scale-Grid gesendet wird, in dem die Authentifizierung und Berechtigung durchgeführt werden.

  1. Konfigurieren Sie die eXtreme-Scale-Clientauthentifizierung und -berechtigung im Grid. Einzelheiten zum Konfigurieren der eXtreme-Scale-Clientauthentifizierung und -berechtigung finden Sie in „Sicherheitsintegration mit externen Providern" auf Seite 375.
  2. Konfigurieren Sie den (vom REST-Service verwendeten) eXtreme-Scale-Client für die Sicherheit.

     Der REST-Datenservice von eXtreme Scale ruft die eXtreme-Scale-Clientbibliothek auf, wenn er mit dem eXtreme-Scale-Grid kommuniziert. Deshalb muss der eXtreme-Scale-Client für die Sicherheit von eXtreme Scale konfiguriert werden.

     Die eXtreme-Scale-Clientauthentifizierung wird über Eigenschaften in der ObjectGrid-Clienteigenschaftendatei aktiviert. Wenn die Clientsicherheit mit den REST-Service verwendet wird, müssen mindestens die folgenden Attribute aktiviert werden:

     ```
     securityEnabled=true
     credentialAuthentication=Supported [-oder-] Required
     credentialGeneratorProps=user:pass [-oder-] {xor encoded user:pass}
     ```

Die mit der Eigenschaft "credentialGeneratorProps" angegebene Benutzer/Kennwort-Kombination muss einer ID im Authentifizierungsregister entsprechen und genügend ObjectGrid-Richtlinienrechte haben, um eine Verbindung zu ObjectGrids herzustellen und ObjectGrids zu erstellen.

Ein Muster für eine ObjectGrid-Clienteigenschaftendatei finden Sie in `wxsrest_home/security/security.ogclient.properties`. Lesen Sie auch den Abschnitt „Clienteigenschaftendatei” auf Seite 207.

3. Konfigurieren Sie den REST-Datenservice von eXtreme Scale für die Sicherheit.

   Die Konfigurationseigenschaftendatei des REST-Datenservice von eXtreme Scale muss für die Integration mit der Sicherheit von eXtreme Scale die folgenden Einträge enthalten:

   ```
   ogClientPropertyFile=Dateiname
   ```

   ogClientPropertyFile gibt die Position der Eigenschaftendatei an, die die im vorherigen Schritt genannten ObjectGrid-Clienteigenschaften enthält. Der REST-Service verwendet diese Datei, um den eXtreme-Scale-Client für die Kommunikation mit dem Grid zu initialisieren, wenn die Sicherheit aktiviert ist.

   ```
   loginType=basic [-oder-] none
   ```

   Die Eigenschaft "loginType" konfiguriert den REST-Service für den Anmeldetyp. Wenn Sie den Wert "none" angeben, wird die mit "credentialGeneratorProps" definierte globale Benutzer-ID/Kennwort-Kombination für jede Transaktion an das Grid gesendet. Wenn Sie den Wert "basic" angeben, sendet der REST-Service eine HTTP-Basisanforderung an den Client und fordert diesen zur Bereitstellung von Berechtigungsnachweisen auf, die bei der Kommunikation mit dem Grid in jeder Transaktion gesendet werden.

   Weitere Informationen zu den Eigenschaften "ogClientPropertyFile" und "loginType" finden Sie unter „Eigenschaftendatei des REST-Datenservice” auf Seite 293.

- Wenden Sie Zugriffsregeln an.

  Der Zugriff kann auch über servicebezogene Konfigurationsregeln, so genannte Zugriffsregeln, gesteuert werden. Es gibt zwei Typen von Zugriffsregeln: Serviceoperationsrechte, die die CRUD-Operationen steuern, die der Service zulässt, und Entitätszugriffsrechte, die die CRUD-Operationen steuern, die für einen bestimmten Entitätstyp zulässig sind.

  Der REST-Datenservice von eXtreme Scale lässt optional Zugriffsregeln zu, die konfiguriert werden können, um den Zugriff auf den Service und die Entitäten im Service zu beschränken. Diese Zugriffsregeln werden in der Eigenschaftendatei mit den Zugriffsrechten für den REST-Datenservice angegeben. Der Name dieser Datei wird, wie in Abschnitt 5.1 gezeigt, in der Eigenschaftendatei des REST-Datenservice mit der Eigenschaft "wxsRestAccessRightsFile" angegeben. Diese Datei ist eine typische Java-Eigenschaftendatei mit Schlüssel/Wert-Paaren. Es gibt zwei Typen von Zugriffsregeln: Serviceoperationsrechte, die die CRUD-Operationen steuern, die der Service zulässt, und Entitätszugriffsrechte, die die CRUD-Operationen steuern, die für einen bestimmten Entitätstyp zulässig sind.

  1. Konfigurieren Sie die Rechte für die Serviceoperationen.

     Die Rechte für Serviceoperationen legen die Zugriffsrechte fest, die für alle ObjectGrids gelten, die über den REST-Service bereitgestellt werden, bzw. für alle Entitäten eines angegebenen ObjectGrids.

     Verwenden Sie die folgende Syntax.

     ```
     serviceOperationRights=Recht_für_Serviceoperation
     serviceOperationRights.Grid-Name -ODER- *=Recht_für_Serviceoperation
     ```

Für diese Angaben gilt Folgendes:

– Die gültigen Werte für "serviceOperationRights2 sind NONE, READSINGLE, READMULTIPLE, ALLREAD und ALL.
– serviceOperationRights.*Grid-Name -ODER-* * impliziert, dass das Zugriffsrecht für alle ObjectGrids gilt, sofern nicht der Name eines bestimmten ObjectGrids angegeben wird.

Beispiel:

```
serviceOperationsRights=ALL
serviceOperationsRights.*=NONE
serviceOperationsRights.EMPLOYEEGRID=READSINGLE
```

Das erste Beispiel gibt an, dass alle Serviceoperationen für alle ObjectGrids ausgeführt werden können, die von diesem REST-Service bereitgestellt werden. Das zweite Beispiel gleicht dem ersten Beispiel insofern, dass es für alle vom REST-Service bereitgestellten ObjectGrids gilt, definiert aber das Zugriffsrecht mit NONE, d. h., dass keine Serviceoperation für die ObjectGrids zulässig ist. Das letzte Beispiel gibt an, wie die Serviceoperationen für ein bestimmtes Grid gesteuert werden. In diesem Fall werden nur "Reads" (Leseoperationen), die einen einzigen Datensatz ergeben, für alle Entitäten des EMPLOYEEGRID zugelassen.

Standardmäßig geht der REST-Service von "serviceOperationsRights=ALL" aus, was bedeutet, dass alle Operationen für alle von diesem Service bereitgestellten ObjectGrids zugelassen werden. Dies unterscheidet sich von der Microsoft-Implementierung, in der der Standardwert NONE ist (d. h. keine Operationen sind im REST-Service zulässig).

Anmerkung: Die Rechte für Serviceoperationen werden in der Reihenfolge ausgewertet, in der sie in dieser Datei angegeben sind, so dass das zuletzt angegebene Recht die vorherigen überschreibt.

2. Konfigurieren Sie die Zugriffsrechte für Entitäten.

   Entitätszugriffsrechte legen die Zugriffsrechte fest, die für bestimmte ObjectGrid-Entitäten gelten, die über den REST-Service bereitgestellt werden. Diese Rechte sind eine Methode für die Einrichtung einer strengeren und differenzierteren Zugriffssteuerung für einzelne ObjectGrid-Entitäten im Vergleich zu Rechten für Serviceoperationen.

   Verwenden Sie die folgende Syntax.

   ```
   entitySetRights.Grid.Name.Entitätsname=Recht_für_Entität
   ```

   Für diese Angaben gilt Folgendes:

   – Die gültigen Werte für *Recht_für_Entität* sind im Folgenden aufgeführt.

Tabelle 18. Entitätszugriffsrechte. Unterstützte Werte

Zugriffsrecht	Beschreibung
NONE	Weist alle Rechte für den Zugriff auf Daten zurück.
READSINGLE	Lässt das Lesen einzelner Datenelemente zu.
READMULTIPLE	Lässt das Lesen von Datengruppen zu.
ALLREAD	Lässt das Lesen einzelner oder mehrerer Datengruppen zu.
WRITEAPPEND	Lässt das Erstellen neuer Datenelemente in Datensätzen zu.
WRITEREPLACE	Lässt das Ersetzen von Daten zu.
WRITEDELETE	Lässt das Löschen von Datenelementen aus Datensätzen zu.
WRITEMERGE	Lässt das Zusammenführen von Daten zu.

Tabelle 18. Entitätszugriffsrechte (Forts.). Unterstützte Werte

Zugriffsrecht	Beschreibung
ALLWRITE	Lässt das Schreiben von Daten zu (d. h. erstellen, ersetzen, zusammenfügen oder löschen).
ALL	Lässt das Erstellen, Lesen, Aktualisieren und Löschen von Daten zu.

  - *Grid-Name* steht für den Namen eines bestimmten ObjectGrids im REST-Service.
  - *Entitätsname* steht für den Namen einer bestimmten Entität im angegebenen ObjectGrid.

  Anmerkung: Wenn Rechte für Serviceoperationen und Entitätzugriffsrechte für ein ObjectGrid und dessen Entitäten festgelegt werden, werden die restriktiveren dieser Rechte umgesetzt, wie in den folgenden Beispielen veranschaulicht wird. Beachten Sie auch, dass die Entitätszugriffsrechte in der Reihenfolge ausgewertet werden, in der sie in der Datei angegeben sind. Das zuletzt angegebene Recht überschreibt die vorherigen Rechte.

  Beispiel 1: Wenn "serviceOperationsRights.NorthwindGrid=READSINGLE" und "entitySetRights.NorthwindGrid.Customer=ALL" angegeben werden, wird READSINGLE für die Entität "Customer" umgesetzt.

  Beispiel 2: Wenn "serviceOperationsRights.NorthwindGrid=ALLREAD" und "entitySetRights.NorthwindGrid.Customer=ALLWRITE" angegeben werden, werden nur Reads (Leseoperationen) für alle Entitäten von NorthwindGrid zugelassen. Für "Customer" verhindern die zugehörigen Entitätszugriffsrechte jedoch alle Leseoperationen (da ALLWRITE angegeben wurde), und somit gilt für die Entität "Customer" das Zugriffsrecht NONE.

- Sichern Sie die Transporte.

  Die Transportsicherheit wird über die Hostcontainerkonfiguration (für Verbindungen vom Webclient zum REST-Service) und über die Clientkonfiguration von eXtreme Scale (für Verbindungen vom REST-Service zum eXtreme-Scale-Grid) bereitgestellt.

  1. Sichern Sie die Verbindung von Client und REST-Service. Die Transportsicherheit für diese Verbindung wird von der Hostcontainerumgebung bereitgestellt und nicht in eXtreme Scale.
  2. Sichern Sie die Verbindung des REST-Service und des eXtreme-Scale-Grids. Die Transportsicherheit für diese Verbindung wird in eXtreme Scale konfiguriert. Weitere Informationen finden Sie unter „Transport Layer Security und Secure Sockets Layer” auf Seite 370.

Kapitel 7. Implementierungsumgebung ausführen

Die Ausführung der Produktumgebung umfasst das Starten und Stoppen von Servern im eigenständigen Modus oder in WebSphere Application Server. Sie können WebSphere eXtreme Scale auch als Sitzungsmanager in einer Umgebung von WebSphere Application Server einsetzen.

Verwaltungsterminologie

Vor der Verwaltung von WebSphere eXtreme Scale sollten Sie sich mit den folgenden Fakten vertraut machen.

Informationen zu diesem Vorgang

Servertypen

In WebSphere eXtreme Scale gibt es zwei Typen von Servern: *Katalogserver* und *Containerserver*. Katalogserver steuern die Verteilung von Shards und erkennen und überwachen die Containerserver. Mehrere Katalogserver zusammen bilden den *Katalogservice*. Containerserver sind die Java Virtual Machines, in denen die Anwendungsdaten für das Grid gespeichert werden.

Eigenständiger Modus

Der eigenständige Modus ist eine Konfiguration von WebSphere eXtreme Scale, die eigenständig, d. h. ohne andere Anwendungsserverprodukte, ausgeführt wird.

Ausführung in WebSphere Application Server

Wenn Sie WebSphere eXtreme Scale in WebSphere Application Server ausführen, werden Katalogserver automatisch in Servern von WebSphere Application Server gestartet. Zum Starten von Containerservern müssen Sie Ihre Anwendung mit ObjectGrid-XML-Dateien packen und implementieren.

Container, Partitionen und Shards

Der Container ist ein Service, der Anwendungsdaten für das Grid speichert. Diese Daten werden gewöhnlich in Teile, so genannte Partitionen, aufgeteilt und auf mehrere Container verteilt. Jeder Container wiederum enthält einen Teil der vollständigen Daten. Eine JVM kann einen oder mehrere Container enthalten und jeder Container mehrere Shards.

Hinweis: Planen Sie die Größe des Heap-Speichers für die Container, in denen alle Ihre Daten enthalten sind. Konfigurieren Sie die Einstellungen für den Heap-Speicher entsprechend.

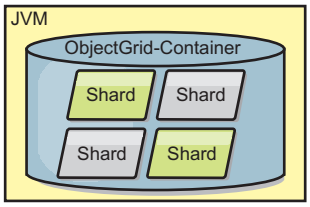


Abbildung 19. Container

Partitionen enthalten einen Teil der Daten des Grids. WebSphere eXtreme Scale stellt automatisch mehrere Partitionen in einen einzigen Container und verteilt die Partitionen dann breiter, wenn weitere Container verfügbar werden.

Wichtig: Sie müssen die Anzahl der Partitionen vor der endgültigen Implementierung sorgfältig auswählen, da sie nicht dynamisch geändert werden kann. Ein Hash-Mechanismus wird verwendet, um Partitionen im Netz zu suchen, und eXtreme Scale hat nach der Implementierung der Daten keine Möglichkeit, ein erneutes Hashing für die gesamten Daten durchzuführen. Als allgemeine Regel gilt, dass die Anzahl der Partitionen zu hoch geschätzt werden kann.

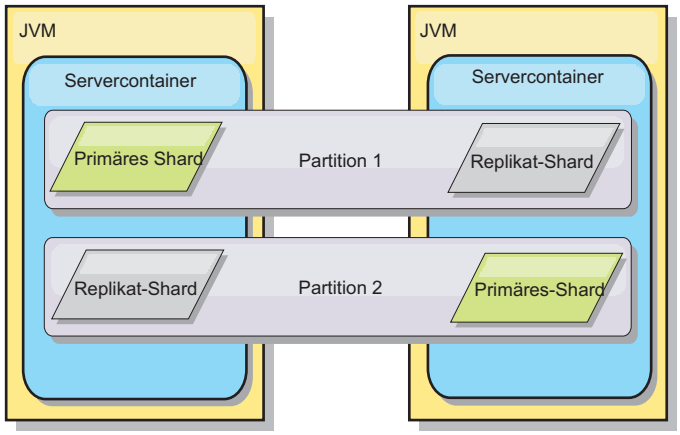


Abbildung 20. Partition

Shards sind Instanzen von Partitionen und haben eine von zwei Rollen: primäres Shard oder Replikat-Shard. Das primäre Shard und seine Replikate bilden die physische Manifestation der Partition. Jede Partition hat mehrere Shards, die jeweils alle in dieser Partition vorhandenen Daten enthalten. Ein Shard ist das primäre Shard, und die anderen Shards sind Replikate oder Replikat-Shards, d. h. redundante Kopien der Daten im primären Shard. Ein primäres Shard ist die einzige Partitionsinstanz, die Transaktionen das Schreiben in den Cache erlaubt. Ein Replikat-Shard ist eine "gespiegelte" Instanz der Partition. Es empfängt synchron oder asynchron Aktualisierungen vom primären Shard. Das Replikat-Shard erlaubt Transaktionen nur das Lesen von Daten aus dem Cache. Replikate befinden sich nie in demselben Container wie das primäre Shard und normalerweise nicht einmal auf derselben Maschine wie das primäre Shard.

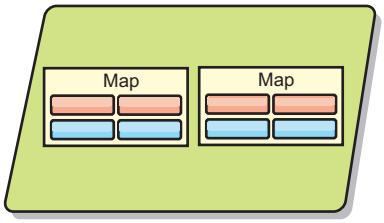


Abbildung 21. Shard

Um die Verfügbarkeit der Daten oder die Persistenzgarantien zu erhöhen, replizieren Sie die Daten. Die Replikation bedeutet jedoch einen höheren Aufwand für die Transaktion und damit Leistungseinbußen zu Gunsten der Verfügbarkeit. Mit eXtreme Scale können Sie den Aufwand steuern, da synchrone und asynchrone Replikation sowie Hybridreplikationsmodelle unterstützt werden, die synchrone und asynchrone Replikationsmodi verwenden. Ein synchrones Replikat-Shard empfängt Aktualisierungen im Rahmen der Transaktion des primären Shards, um die Datenkonsistenz zu gewährleisten. Ein synchrones Replikat kann die Antwortzeit verdoppeln, da die Transaktion sowohl im primären Shard als auch im synchronen Replikat festgeschrieben werden muss, bevor sie beendet werden kann. Ein asynchrones Replikat-Shard empfängt Aktualisierungen nach der Transaktionsfestschreibung, um die Auswirkungen auf die Leistung zu begrenzen, birgt aber das Risiko eines Datenverlusts, da das asynchrone Replikat mehrere Transaktionen hinter dem primären Shard zurückliegen kann.

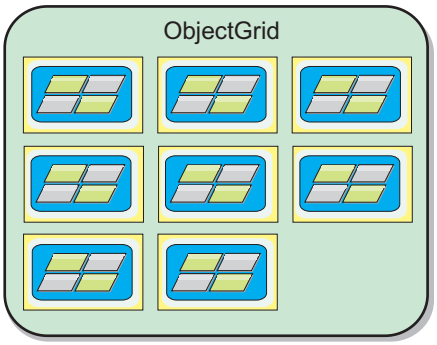


Abbildung 22. ObjectGrid

Katalogservices (Katalogserver)

Der Katalogservice enthält Logik, die bei stabilem Zustand der Topologie inaktiv bleibt und nur geringen Einfluss auf die Skalierbarkeit hat. Der Katalogservice ist so konzipiert, dass er Hunderte gleichzeitig verfügbarer Container bedienen kann, und führt Services für die Verwaltung der Container aus.

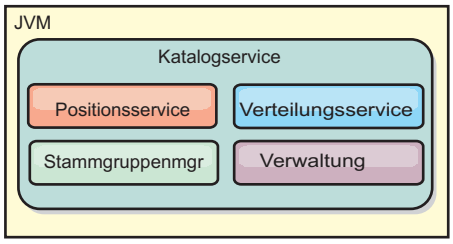


Abbildung 23. Katalogservice

Die Zuständigkeiten des Katalogs setzen sich aus den folgenden Services zusammen:

Arbeitsumgebung
Der Positionsservice stellt Positionen für Clients, die Container mit Anwendungen suchen, und für Container, die bereitgestellte Anwendungen beim Verteilungsservice registrieren möchten, bereit. Der Positionsservice wird zur horizontalen Skalierung dieser Funktion in allen Grid-Membern ausgeführt.

Verteilungsservice
Der Verteilungsservice ist ein zentrales "Nervensystem" für das Grid und für die Zuordnung einzelner Shards zu ihrem Hostcontainer verantwortlich. Der Verteilungsservice wird als einer von N ausgewählten Services im Cluster ausgeführt. Da die Eins-von-N-Richtlinie verwendet wird, ist immer genau eine Instanz des Verteilungsservice aktiv. Wenn diese Instanz gestoppt wird, übernimmt ein anderer Prozess ihre Arbeit. Alle Zustände des Katalogservice werden für die Redundanz auf allen Servern repliziert, die den Katalogservice enthalten.

Stammgruppenmanager
Der Stammgruppenmanager verwaltet die Peer-Gruppierung für die Vitalitätsüberwachung, fasst Container zu kleinen Servergruppen zusammen und bindet die Servergruppen automatisch ein. Wenn ein Container den ersten Kontakt zum Katalogservice herstellt, wartet der Container auf die Zuordnung zu einer neuen oder vorhandenen Gruppe mehrerer Java Virtual Machines (JVM). Jede JVM-Gruppe überwacht die Verfügbarkeit ihrer Member durch den Austausch von Überwachungssignalen. Eines der Gruppen-Member übermittelt dem Katalogservice die Verfügbarkeitsdaten, damit dieser durch Neuzuordnung und Routenweiterleitung auf Fehler reagieren kann.

Verwaltung
Die vier Phasen der Verwaltung einer Umgebung mit WebSphere eXtreme Scale sind Planung, Implementierung, Verwaltung und Überwachung. Weitere Informationen zu jeder dieser Phasen finden Sie im *Administratorhandbuch*.

Für die Verfügbarkeit konfigurieren Sie eine Katalogservicedomäne. Eine Katalogservicedomäne setzt sich aus mehreren Java Virtual Machines, einschließlich einer Master-JVM und einer Reihe von Ausweich-JVMs zusammen.

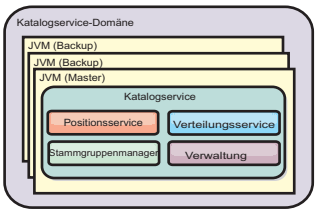


Abbildung 24. Katalogservicedomäne

Verfügbarkeit eines ObjectGrids festlegen

Der Verfügbarkeitsstatus einer ObjectGrids-Instanz bestimmt, welche Anforderungen zu einer bestimmten Zeit verarbeitet werden können.

Es gibt vier Verfügbarkeitsstatus:

- ONLINE
- QUIESCE
- OFFLINE
- PRELOAD

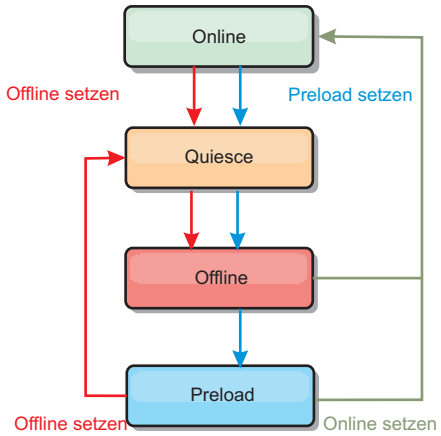


Abbildung 25. Verfügbarkeitsstatus eines ObjectGrids

Verfügbarkeitsstatus festlegen

Standardmäßig hat ein ObjectGrid den Verfügbarkeitsstatus ONLINE. Ein ObjectGrid mit dem Verfügbarkeitsstatus ONLINE kann alle Anforderungen eines typischen eXtreme-Scale-Clients verarbeiten. Anforderungen von Preload-Clients werden jedoch zurückgewiesen, wenn das ObjectGrid den Status ONLINE hat.

Der Status QUIESCE (Stilllegen) ist ein Übergangsstatus. Ein ObjectGrid mit dem Verfügbarkeitsstatus QUIESCE wird bald in den Status OFFLINE versetzt. Wenn ein ObjectGrid den Status QUIESCE hat, können ausstehende Transaktionen verarbeitet werden. Neue Transaktionen werden jedoch zurückgewiesen. Ein ObjectGrid kann bis zu 30 Sekunden im Status QUIESCE verbleiben. Danach wird der Verfügbarkeitsstatus in OFFLINE geändert.

Ein ObjectGrid im Status OFFLINE weist alle Transaktionen zurück.

Der Status PRELOAD kann verwendet werden, um Daten von einem Preload-Client in ein ObjectGrid zu laden. Während das ObjectGrid den Status PRELOAD hat, kann nur ein Preload-Client Transaktionen im ObjectGrid festschreiben. Alle anderen Transaktionen werden zurückgewiesen.

Verwenden Sie die Schnittstelle "StateManager", um den Verfügbarkeitsstatus eines ObjectGrids festzulegen. Zum Festlegen des Verfügbarkeitsstatus eines auf den Servern ausgeführten ObjectGrids übergeben Sie einen entsprechenden ObjectGrid-Client an die Schnittstelle "StateManager". Der folgende Code veranschaulicht, wie der Verfügbarkeitsstatus eines ObjectGrids geändert wird.

```
ClientClusterContext client = ogManager.connect("localhost:2809", null, null);
ObjectGrid myObjectGrid = ogManager.getObjectGrid(client, "myObjectGrid");
StateManager stateManager = StateManagerFactory.getStateManager();
stateManager.setObjectGridState(AvailabilityState.OFFLINE, myObjectGrid);
```

Jedes Shard des ObjectGrids nimmt den gewünschten Status an, wenn die Methode "setObjectGridState" in der Schnittstelle "StateManager" aufgerufen wird. Wenn die Methode zurückkehrt, sollten alle Shards im ObjectGrid den richtigen Status haben.

Verwenden Sie ein ObjectGridEventListener-Plug-in, um den Verfügbarkeitsstatus eines serverseitigen ObjectGrids zu ändern. Ändern Sie den Verfügbarkeitsstatus eines serverseitigen ObjectGrids nur, wenn das ObjectGrid eine einzige Partition hat. Falls das ObjectGrid mehrere Partitionen hat, wird die Methode "shardActivated" für jedes primäre Shard aufgerufen, was zu überflüssigen Aufrufen zum Ändern des ObjectGrid-Status führt.

```
public class OGListener implements ObjectGridEventListener,
   ObjectGridEventGroup.ShardEvents {
    public void shardActivated(ObjectGrid grid) {
            StateManager stateManager = StateManagerFactory.getStateManager();
            stateManager.setObjectGridState(AvailabilityState.PRELOAD, grid);
    }
}
```

Da der Status QUIESCE ein Übergangsstatus ist, können Sie die Schnittstelle "StateManager" nicht verwenden, um ein ObjectGrid in den Status QUIESCE zu versetzen. Ein ObjectGrid nimmt diesen Status auf seinem Weg zum Status OFFLINE vorübergehend an.

Verfügbarkeitsstatus abrufen

Verwenden Sie die Methode "getObjectGridState" der Schnittstelle "StateManager", um den Verfügbarkeitsstatus eines bestimmten ObjectGrids abzurufen.

```
StateManager stateManager = StateManagerFactory.getStateManager();
AvailabilityState state = stateManager.getObjectGridState(inventoryGrid);
```

Die Methode "getObjectGridState" wählt ein zufälliges primäres Shard im ObjectGrid aus und gibt dessen Verfügbarkeitsstatus zurück. Da alle Shards eines ObjectGrids denselben Verfügbarkeitsstatus haben bzw. auf dem Übergang zu demselben Verfügbarkeitsstatus sein sollten, gibt diese Methode ein akzeptables Ergebnis für den aktuellen Verfügbarkeitsstatus des ObjectGrids zurück.

Erforderliche Verfügbarkeitsstatus für verschiedene Anforderungen

Eine Anforderung wird zurückgewiesen, wenn ein ObjectGrid nicht den erforderlichen Verfügbarkeitsstatus hat, der diese Anforderung unterstützt. Es wird eine Ausnahme des Typs "AvailabilityException" ausgegeben, wenn eine Anforderung zurückgewiesen wird.

Attribut "initialState"

Sie können das Attribut "initialState" in einem ObjectGrid verwenden, um dessen Anfangsstatus anzugeben. Nach Abschluss der Initialisierung ist ein ObjectGrid normalerweise für Routing bereit. Der Status kann später geändert werden, um zu verhindern, dass Datenverkehr an ein ObjectGrid weitergeleitet wird. Wenn das ObjectGrid initialisiert werden muss, aber nicht sofort verfügbar ist, können Sie das Attribut "initialState" verwenden.

Das Attribut "initialState" wird in der XML-Konfigurationsdatei des ObjectGrids definiert. Der Standardstatus ist ONLINE. Die gültigen Werte sind:

- ONLINE (Standardeinstellung)
- PRELOAD
- OFFLINE

Weitere Informationen finden Sie in der API-Dokumentation zu AvailabilityState.

Wenn das Attribut "initialState" in einem ObjectGrid definiert wird, muss der Status explizit auf ONLINE zurückgesetzt werden, oder das ObjectGrid bleibt nicht verfügbar. Es werden Ausnahmen des Typs "AvailabilityException" ausgelöst.

Attribut "initialState" für das vorherige Laden verwenden

Wenn das ObjectGrid vorher mit Daten geladen wird (Preload), kann es einen Zeitraum zwischen Verfügbarkeit des ObjectGrids und Wechseln in einen Preload-Status geben, in dem Clientdatenverkehr blockiert werden kann. Um diesen Zeitraum zu verhindern, kann der Anfangsstatus eines ObjectGrids auf PRELOAD gesetzt werden. Das ObjectGrid setzt zwar die erforderlichen Initialisierungsprozesse fort, blockiert den Datenverkehr aber so lange, bis sich der Status ändert und der Preload-Prozess durchgeführt werden kann.

Die Status PRELOAD und OFFLINE blockieren zwar beide den Datenverkehr, aber Sie müssen den Status PRELOAD verwenden, wenn Sie einen Preload-Prozess einleiten möchten.

Verhalten beim Failover und Lastausgleich

Wenn ein Replikat in ein primäres Shard hochgestuft wird, verwendet es nicht die initialState-Einstellung. Wenn das primäre Shard zur Neuverteilung verschoben wird, wird die initialState-Einstellung nicht verwendet, weil die Daten an die neue primäre Position kopiert werden, bevor der Verschiebevorgang durchgeführt wird. Wenn keine Replikation konfiguriert ist, übernimmt das primäre Shard den initialState-Wert, wenn ein Failover stattfindet und ein neues primäres Shards verteilt werden muss.

Integrierte Server-API verwenden

Mit WebSphere eXtreme Scale können Sie eine programmgesteuerte API verwenden, um den Lebenszyklus integrierter Server und Container zu verwalten. Sie können den Server über das Programm mit jeder der Optionen konfigurieren, die Sie auch über die Befehlszeilenoptionen oder dateibasierten Servereigenschaften konfigurieren können. Sie können den integrierten Server als Containerserver und/oder Katalogservice konfigurieren.

Vorbereitende Schritte

Sie müssen eine Methode für die Ausführung von Code über eine bereits vorhandene Java Virtual Machine haben. Die eXtreme-Scale-Klassen müssen über die Baumstruktur der Klassenladeprogramme verfügbar sein.

Informationen zu diesem Vorgang

Viele Verwaltungs-Tasks können über die Verwaltungs-API ausgeführt werden. Die API wird häufig als interner Server für die Speicherung des Webanwendungsstatus eingesetzt. Der Webserver kann als integrierter WebSphere eXtreme Scale-Server gestartet werden, den Containerserver dem Katalogservice melden, und anschließend wird der Server als Member eines größeren verteilten Grids hinzugefügt. Diese Verwendung kann aus einem ansonsten flüchtigen Datenspeicher einen skalierbaren und hoch verfügbaren Datenspeicher machen.

Sie können den vollständigen Lebenszyklus eines integrierten eXtreme-Scale-Servers über das Programm steuern. Die Beispiele sind so generisch wie möglich und enthalten nur direkte Codemuster für die beschriebenen Schritte.

Vorgehensweise

1. Rufen Sie das Objekt "ServerProperties" aus der Klasse "ServerFactory" ab, und konfigurieren Sie alle erforderlichen Optionen.

   Jeder eXtreme-Scale-Server besitzt eine Reihe konfigurierbarer Eigenschaften. Wenn ein Server über die Befehlszeile gestartet wird, werden diese Eigenschaften auf Standardwerte gesetzt, aber Sie können mehrere Eigenschaften überschreiben, indem Sie eine externe Quelle oder Datei angeben. Im integrierten Bereich können Sie die Eigenschaften direkt mit einem ServerProperties-Objekt setzen. Sie müssen diese Eigenschaften setzen, bevor Sie eine Serverinstanz aus der Klasse "ServerFactory" abrufen. Das folgende Beispiel-Snippet ruft ein ServerProperties-Objekt ab, setzt das Feld "CatalogServiceBootStrap" und initialisiert mehrere optionale Servereinstellungen. Eine Liste der konfigurierbaren Einstellungen finden Sie in der API-Dokumentation.

   ```
   ServerProperties props = ServerFactory.getServerProperties();
   props.setCatalogServiceBootstrap("host:port");
   // Für Verbindungsherstellung zu einem bestimmtem Katalogserver erforderlich
   props.setServerName("ServerOne"); // Server benennen
   props.setTraceSpecification("com.ibm.ws.objectgrid=all=enabled"); // Trace-Spezifikation festlegen
   ```

2. Wenn der Server ein Katalogserver sein soll, rufen Sie das Objekt "CatalogServerProperties" ab.

   Jeder integrierte Server kann ein Katalogserver und/oder ein Containerserver sein. Der folgende Beispielcode ruft das Objekt "CatalogServerProperties" ab, aktiviert die Katalogserviceoption und konfiguriert verschiedene Einstellungen des Katalogservice.

   ```
   CatalogServerProperties catalogProps = ServerFactory.getCatalogProperties();
   catalogProps.setCatalogServer(true);
   // standardmäßig false; erforderlich für die Einstellung als Katalogservice
   catalogProps.setQuorum(true); // Quorum aktivieren/inaktivieren
   ```

3. Rufen Sie eine Server-Instanz aus der Klasse "ServerFactory" ab. Die Server-Instanz ist ein prozessbezogenes Singleton, das für die Verwaltung der Zugehörigkeiten im Grid zuständig ist. Nach der Instanziierung dieser Instanz ist dieser Prozess verbunden und zusammen mit den anderen Servern im Grid hoch verfügbar. Das folgende Beispiel veranschaulicht, wie die Server-Instanz erstellt wird:

   ```
   Server server = ServerFactory.getInstance();
   ```

   Wenn Sie sich das vorherige Beispiel ansehen, stellen Sie fest, dass die Klasse "ServerFactory" eine statische Methode bereitstellt, die eine Server-Instanz zurückgibt. Die Klasse "ServerFactory" ist die einzige geplante Schnittstelle für das Abrufen einer Server-Instanz. Deshalb stellt die Klasse sicher, dass die Instanz ein Singleton bzw. die einzige Instanz für jede JVM bzw. jedes isolierte Klassenladeprogramm ist. Die Methode "getInstance" initialisiert die Server-Instanz. Sie müssen alle Servereigenschaften konfigurieren, bevor Sie die Instanz initialisieren. Die Klasse "Server" ist für die Erstellung neuer Container-Instanzen zuständig. Sie können die Klassen "ServerFactory" und "Server" verwenden, um den Lebenszyklus der integrierten Serverinstanz zu verwalten.

4. Starten Sie eine Container-Instanz über die Serverinstanz.

   Bevor Shards an einen integrierten Server verteilt werden können, müssen Sie einen Container im Server erstellen. Die Schnittstelle "Server" besitzt eine Methode "createContainer", die das Argument "DeploymentPolicy" akzeptiert. Im folgenden Beispiel wird die Serverinstanz, die Sie zum Erstellen eines Containers abgerufen haben, mit einer erstellten DeploymentPolicy-Datei verwendet. Beachten Sie, dass Container für die Serialisierung ein Klassenladeprogramm erfordern, dem die Binärdateien der Anwendung zur Verfügung stehen. Sie können diese Binärdateien bereitstellen, indem Sie die Methode "createContainer" aufrufen und in diesem Aufruf das Klassenladeprogramm für den Thread-Kontext auf das Klassenladeprogramm setzen, das Sie verwenden möchten.

   ```
   DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy(new
      URL("file://urltodeployment.xml"),
    new URL("file://urltoobjectgrid.xml"));
   Container container = server.createContainer(policy);
   ```

5. Entfernen und bereinigen Sie einen Container.

   Sie können einen Containerserver entfernen und bereinigen, indem Sie die Methode "teardown" für die abgerufene Containerinstanz ausführen. Bei der Ausführung der Methode "teardown" für einen Container wird der Container bereinigt und aus dem integrierten Server entfernt.

   Die Bereinigung des Containers beinhaltet die Verlagerung und Umrüstung aller Shards dieses Containers. Jeder Server kann mehrere Container und Shards enthalten. Die Bereinigung eines Containers hat keine Auswirkung auf den Lebenszyklus der übergeordneten Server-Instanz. Das folgende Beispiel veranschaulicht, wie die Methode "teardown" für einen Server ausgeführt wird. Die Methode "teardown" wird über die Schnittstelle "ContainerMBean" bereitgestellt. Wenn Sie keinen Zugriff mehr über das Programm auf diesen Container haben und die Schnittstelle "ContainerMBean" verwenden, können Sie den Container mit der zugehörigen MBean trotzdem bereinigen. Die Schnittstelle "Container" enthält auch eine Methode "terminate". Verwenden Sie diese Methode nur, wenn es unbedingt erforderlich ist. Diese Methode ist konsequenter und koordiniert die entsprechende Shard-Verlagerung und -Bereinigung nicht.

   ```
   container.teardown();
   ```

6. Stoppen Sie den integrierten Server.

   Wenn Sie einen integrierten Server stoppen, stoppen Sie auch alle Container und Shards, die im Server ausgeführt werden. Wenn Sie einen integrierten Server stoppen, müssen Sie alle offenen Verbindungen bereinigen und alle Shards

verlagern oder umrüsten. Das folgende Beispiel veranschaulicht, wie ein Server gestoppt wird und die Methode "waitFor" in der Schnittstelle "Server" verwendet wird, um sicherzustellen, dass die Serverinstanz vollständig beendet wird. Ähnlich wie im Containerbeispiel wird die Methode "stopServer" über die Schnittstelle "ServerMBean" bereitgestellt. Mit dieser Schnittstelle können Sie einen Server über die entsprechende Managed Bean (MBean) stoppen.

```
ServerFactory.stopServer();  // Factory zum Beenden des Server-Singletons
 // oder
server.stopServer(); // Serverinstanz direkt verwenden
server.waitFor();
// Kehrt zurück, wenn die Beendigungsprozedur für den Server ordnungsgemäß abgeschlossen wurde
```

Vollständiges Codebeispiel:

```
 import java.net.MalformedURLException;
import java.net.URL;

import com.ibm.websphere.objectgrid.ObjectGridException;
import com.ibm.websphere.objectgrid.deployment.DeploymentPolicy;
import com.ibm.websphere.objectgrid.deployment.DeploymentPolicyFactory;
import com.ibm.websphere.objectgrid.server.Container;
import com.ibm.websphere.objectgrid.server.Server;
import com.ibm.websphere.objectgrid.server.ServerFactory;
import com.ibm.websphere.objectgrid.server.ServerProperties;

public class ServerFactoryTest {

    public static void main(String[] args) {

        try {

            ServerProperties props = ServerFactory.getServerProperties();
            props.setCatalogServiceBootstrap("catalogservice-hostname:catalogservice-port");
            props.setServerName("ServerOne"); // Server benennen
            props.setTraceSpecification("com.ibm.ws.objectgrid=all=enabled"); // TraceSpec

            /*
             * In den meisten Fällen dient der Server nur als Containerserver und
             * stellt eine Verbindung zu einem externen Katalogserver her. Auf diese
             * Weise wird eine höhere Verfügbarkeit erreicht. Der folgende, auf Kommentar
             * gesetze Codeauszug aktiviert diesen Server als Katalogserver.
             *
             *
             * CatalogServerProperties catalogProps =
             * ServerFactory.getCatalogProperties();
             * catalogProps.setCatalogServer(true); // Katalogservice aktivieren
             * catalogProps.setQuorum(true); // Quorum aktivieren
             */

            Server server = ServerFactory.getInstance();

            DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy
     (new URL("url to deployment xml"),  new URL("url to objectgrid xml file"));
            Container container = server.createContainer(policy);

            /*
             * Das Shard wird jetzt an diesen Container verteilt, wenn die
             * Implementierungsanforderungen erfüllt sind.
             * Dies umfasst die Erstellung des integrierten Servers und des Containers.
             *
             * Die folgenden Zeilen demonstrieren lediglich den Aufruf der Bereinigungsmethoden.
             */

            container.teardown();
            server.stopServer();
            int success = server.waitFor();

        } catch (ObjectGridException e) {
            // Container konnte nicht initialisiert werden.
        } catch (MalformedURLException e2) {
            // Ungültiger URL für XML-Datei(en)
        }

    }

}
```

Integrierte Server-API

WebSphere eXtreme Scale enthält Anwendungsprogrammierschnittstellen (APIs) und Systemprogrammierschnittstellen für die Integration von eXtreme-Scale-Servern und -Clients in vorhandenen Java-Anwendungen. Im folgenden Abschnitt werden die verfügbaren integrierten Server-APIs beschrieben.

eXtreme-Scale-Server instanziieren

Sie können verschiedene Eigenschaften verwenden, um die eXtreme-Scale-Serverinstanz zu konfigurieren, die Sie mit der Methode "ServerFactory.getServerProperties" abrufen können. Das ServerProperties-Objekt ist das Singleton, und deshalb ruft jeder Aufruf der Methode "getServerProperties" dieselbe Instanz ab.

Sie können einen neuen Server mit dem folgenden Code erstellen.

```
Server server = ServerFactory.getInstance();
```

Alle Eigenschaften, die vor ersten Aufruf von "getInstance" gesetzt werden, werden zum Initialisieren des Servers verwendet.

Servereigenschaften abrufen

Sie können die Servereigenschaften festlegen, bis die Methode "ServerFactory.getInstance" zum ersten Mal aufgerufen wird. Der erste Aufruf der Methode "getInstance" instanziiert den eXtreme-Scale-Server und liest alle konfigurierten Eigenschaften. Das Festlegen von Eigenschaften nach der Erstellung der Instanz hat keine Auswirkung. Das folgende Beispiel zeigt, wie Eigenschaften vor der Instanziierung einer Server-Instanz definiert werden.

```
// Servereigenschaften abrufen, die diesem Prozess zugeordnet sind.
ServerProperties serverProperties = ServerFactory.getServerProperties();

// Servernamen für diesen Prozess festlegen.
serverProperties.setServerName("EmbeddedServerA");

// Namen der Zone festlegen, in der sich dieser Prozess befindet.
serverProperties.setZoneName("EmbeddedZone1");

// Erforderliche Endpunktinformationen zum Booten des Katalogservice festlegen.
serverProperties.setCatalogServiceBootstrap("localhost:2809");

// Hostnamen des ORB-Listeners festlegen, zu dem die Bindung hergestellt werden soll.
serverProperties.setListenerHost("host.local.domain");

// ORB-Listener-Port festlegen, zu dem die Bindung hergestellt werden soll.
serverProperties.setListenerPort(9010);

// Alle MBeans für diesen Prozess inaktivieren.
serverProperties.setMBeansEnabled(false);

Server server = ServerFactory.getInstance();
```

Integrierter Katalogservice

Jede JVM, die von der Methode "CatalogServerProperties.setCatalogServer" markiert wird, kann den Katalogservice für eXtreme Scale ausführen. Diese Methode weist die Laufzeitumgebung des eXtreme-Scale-Servers an, den Katalogservice zu instanziieren, wenn der Server gestartet wird. Der folgende Code veranschaulicht, wie der Katalogserver von eXtreme Scale instanziiert wird:

```
CatalogServerProperties catalogServerProperties =
 ServerFactory.getCatalogProperties();
catalogServerProperties.setCatalogServer(true);

Server server = ServerFactory.getInstance();
```

eXtreme-Scale-Container integrieren

Setzen Sie die Methode "Server.createContainer" für jede JVM ab, die mehrere eXtreme-Scale-Container ausführen soll. Der folgende Code veranschaulicht, wie ein eXtreme-Scale-Container instanziiert wird:

```
Server server = ServerFactory.getInstance();
DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy(
    new File("META-INF/embeddedDeploymentPolicy.xml").toURI().toURL(),
    new File("META-INF/embeddedObjectGrid.xml").toURI().toURL());
Container container = server.createContainer(policy);
```

Eigenständiger Serverprozess

Sie können alle Services gemeinsam starten, was für die Entwicklung hilfreich und auch in einer Produktionsumgebung praktisch ist. Wenn die Services gemeinsam gestartet werden, führt ein einziger Prozess alle folgenden Aufgaben aus: Er startet den Katalogservice, er startet eine Gruppe von Containern und er führt die Clientverbindungslogik aus. Mit dieser Art des Servicestarts können Programmierungsprobleme festgestellt werden, bevor die Services in einer verteilten Umgebung implementiert werden. Der folgende Code veranschaulicht, wie ein eigenständiger eXtreme-Scale-Server instanziiert wird:

```
CatalogServerProperties catalogServerProperties =
 ServerFactory.getCatalogProperties();
catalogServerProperties.setCatalogServer(true);

Server server = ServerFactory.getInstance();
DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy(
    new File("META-INF/embeddedDeploymentPolicy.xml").toURI().toURL(),
    new File("META-INF/embeddedObjectGrid.xml").toURI().toURL());
Container container = server.createContainer(policy);
```

eXtreme Scale in WebSphere Application Server integrieren

Die Konfiguration für eXtreme Scale wird automatisch vorgenommen, wenn Sie WebSphere Extended Deployment DataGrid in einer Umgebung mit WebSphere Application Server installieren. Sie müssen keine Eigenschaften festlegen, bevor Sie auf den Server zugreifen, um einen Container zu erstellen. Der folgende Code veranschaulicht, wie Sie einen eXtreme-Scale-Server in WebSphere Application Server instanziieren:

```
Server server = ServerFactory.getInstance();
DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy(
    new File("META-INF/embeddedDeploymentPolicy.xml").toURI().toURL(),
    new File("META-INF/embeddedObjectGrid.xml").toURI().toURL);
Container container = server.createContainer(policy);
```

Ein schrittweises Beispiel für das Starten eines integrierten Katalogservice und -containers über das Programm finden Sie unter „Integrierte Server-API verwenden” auf Seite 330.

Eigenständige Server von WebSphere eXtreme Scale starten

Wenn Sie eine eigenständige Konfiguration von WebSphere eXtreme Scale verwenden, setzt sich die Umgebung aus Katalogservern, Containerservern und eXtreme-Scale-Clientprozessen zusammen. Server von eXtreme Scale können mit Hilfe der integrierten Server-API auch in vorhandene Java-Anwendungen integriert werden. Sie müssen diese Prozesse manuell konfigurieren und starten.

Vorbereitende Schritte

Sie können Server von WebSphere eXtreme Scale in einer Umgebung ohne WebSphere Application Server starten. Wenn Sie WebSphere Application Server verwenden, lesen Sie den Abschnitt „WebSphere eXtreme Scale mit WebSphere Application Server verwalten” auf Seite 348.

Nächste Schritte

Stoppen Sie Ihre eXtreme-Scale-Prozesse. Weitere Informationen finden Sie im Abschnitt „Eigenständige eXtreme-Scale-Server stoppen” auf Seite 344.

Katalogservice in einer eigenständigen Umgebung starten

Sie müssen den Katalogservice manuell starten, wenn Sie eine verteilte Umgebung von WebSphere eXtreme Scale ohne WebSphere Application Server verwenden.

Vorbereitende Schritte

Wenn Sie WebSphere Application Server verwenden, wird der Katalogservice automatisch in einem der vorhandenen Prozesse gestartet. Weitere Informationen finden Sie im Abschnitt "Katalogservice in WebSphere Application Server starten".

Informationen zu diesem Vorgang

Der Katalogservice kann in einem einzelnen Prozess ausgeführt werden, oder es können mehrere Katalogserver zur Katalogservicedomäne zusammengefasst werden. Ein Katalogserver-Grid ist in einer Produktionsumgebung für hohe Verfügbarkeit erforderlich. Weitere Informationen zum Konfigurieren einer Katalogservicedomäne finden Sie in den Informationen zu Katalogservicedomänen in der *Produktübersicht*. Der Katalogservice wird, unabhängig davon, ob er in einem Grid oder in einem einzelnen Prozess ausgeführt wird, über das Script `startOgServer` gestartet. Außerdem können Sie zusätzliche Scriptparameter angeben, um den ORB (Object Request Broker) an eine bestimmte Host/Port-Kombination zu binden, die Domäne anzugeben oder die Sicherheit zu aktivieren.

Wenn Sie den Startbefehl aufrufen, verwenden Sie auf UNIX-Plattformen das Script `startOgServer.sh` und unter Windows das Script `startOgServer.bat`.

Vorgehensweise

- **Einzelnen Katalogserverprozess starten**

  Geben Sie zum Starten eines einzelnen Katalogservers die folgenden Befehle in der Befehlszeile ein:

  1. Navigieren Sie wie folgt zum Verzeichnis "bin":
     ```
     cd ObjectGrid-Stammverzeichnis/bin
     ```
  2. Führen Sie den Befehl "startOgServer" aus:
     ```
     startOgServer.bat|sh catalogServer
     ```

Eine Liste aller verfügbaren Befehlszeilenparameter finden Sie im Abschnitt „Script "startOgServer"" auf Seite 340. Wenn Sie den Katalogservice in einer Produktionsumgebung ausführen, verwenden Sie keine einzelne Java Virtual Machine (JVM). Wenn der Katalogservice fehlschlägt, können in diesem Fall keine neuen Clients Anforderungen an die implementierte eXtreme-Scale-Umgebung weitergeleitet werden, und es können keine neuen ObjectGrid-Instanzen zur Domäne hinzugefügt werden. Aus diesen Gründen sollten Sie eine Gruppe von Java Virtual Machines starten, um eine Katalogservicedomäne auszuführen.

- **Katalogservicedomäne mit mehreren Prozessen starten**

  Wenn Sie eine Gruppe von Servern zum Ausführen eines Katalogservice starten möchten, müssen Sie die Option **-catalogServiceEndPoints** im Script `startOgServer` verwenden. Dieses Argument akzeptiert eine Liste mit Katalogserviceendpunkten im Format `Servername:Hostname:Clientport:Peer-Port`. Die Attribute sind wie folgt definiert:

  serverName
  Gibt einen Namen an, der den Prozess identifiziert, den Sie starten.

  hostName
  Gibt den Hostnamen des Computers an, auf dem der Server gestartet wird.

  clientPort
  Gibt den Port an, der für die Peer-Kommunikation im Katalog-Grid verwendet wird.

  peerPort
  Entspricht dem haManagerPort. Gibt den Port an, der für die Peer-Kommunikation im Katalog-Grid verwendet wird.

  Im folgenden Beispiel wird gezeigt, wie die erste von drei Java Virtual Machines für einen Katalogservice gestartet wird:

  1. Navigieren Sie wie folgt zum Verzeichnis "bin":
     ```
     cd ObjectGrid-Stammverzeichnis/bin
     ```
  2. Führen Sie den Befehl "startOgServer" aus:
     ```
     startOgServer.bat|sh cs1 -catalogServiceEndpoints
     cs1:MyServer1.company.com:6601:6602,
     cs2:MyServer2.company.com:6601:6602,
     cs3:MyServer3.company.com:6601:6602
     ```

     In diesem Beispiel wird der Server `cs1` auf dem Host `MyServer1.company.com` gestartet. Dieser Servername ist das erste Argument, das an das Script übergeben wird. Während der Initialisierung von Server `cs1` werden die catalogServiceEndpoints-Parameter untersucht, um die für diesen Prozess zugeordneten Ports zu bestimmen. Die Liste wird auch verwendet, um Server `cs1` das Annehmen von Verbindungen von anderen Servern (`cs2` und `cs3`) zu ermöglichen.
  3. Zum Starten der verbleibenden Katalogserver in der Liste, übergeben Sie die folgenden Argumente an das Script "startOgServer". Es soll der Server `cs2` auf dem Host `MyServer2.company.com` gestartet werden:
     ```
     startOgServer.bat|sh cs2 -catalogServiceEndpoints
     cs1:MyServer1.company.com:6601:6602,
     cs2:MyServer2.company.com:6601:6602,
     cs3:MyServer3.company.com:6601:6602
     ```

     Es soll der Server `cs3` auf dem Host `MyServer3.company.com` gestartet werden:

```
startOgServer.bat|sh cs3 -catalogServiceEndpoints
cs1:MyServer1.company.com:6601:6602,
cs2:MyServer2.company.com:6601:6602,
cs3:MyServer3.company.com:6601:6602
```

Wichtig: Starten Sie alle Katalogserver parallel.

Sie müssen Katalogserver, die in einem Grid enthalten sind, parallel starten, weil jeder Server wartet, bis die anderen Katalogserver der Stammgruppe beitreten. Ein für ein Grid konfigurierter Katalogserver wird erst gestartet, wenn er die anderen Member in der Gruppe identifiziert. Der Katalogserver überschreitet das zulässige Zeitlimit, wenn keine anderen Server verfügbar werden.

- **ORB an eine bestimmte Host/Port-Kombination binden**

  Neben den Ports, die mit dem Argument **catalogServiceEndpoints** definiert werden, verwendet jeder Katalogservice einen Object Request Broker (ORB), um Verbindungen von Clients und Containern zu akzeptieren. Standardmäßig ist der ORB an Port `2809` des lokalen Hosts empfangsbereit. Wenn Sie den ORB an eine bestimmte Host/Port-Kombination in der JVM des Katalogservice binden möchten, verwenden Sie dazu die Argumente **-listenerHost** und **-listenerPort**. Im folgenden Beispiel wird gezeigt, wie Sie einen Katalogserver mit einer einzelnen JVM starten und den zugehörigen ORB an Port 7000 auf MyServer1.company.com binden:

  ```
  startOgServer.sh catalogServer -listenerHost MyServer1.company.com
  -listenerPort 7000
  ```

  Jedem eXtreme-Scale-Container und -Client müssen die ORB-Endpunktdaten des Katalogservice bereitgestellt werden. Clients benötigen nur einen Teil dieser Daten, aber Sie sollten für eine hohe Verfügbarkeit mindestens zwei Endpunkte verwenden.

- **Domäne benennen**

  Ein Domänenname ist nicht erforderlich, wenn Sie einen Katalogservice starten. Der Standarddomänenname ist `defaultDomain`. Wenn Sie Ihre Domäne benennen möchten, verwenden Sie die Option **-domain**. Im folgenden Beispiel wird demonstriert, wie Sie einen Katalogservice mit einer einzelnen JVM und dem Domänennamen `myDomain` starten.

  ```
  startOgServer.sh catalogServer -domain myDomain
  ```

- **Sicheren Katalogservice starten**

  Sie können einen sicheren Katalogservice starten, indem Sie die folgenden Argumente angeben:

  -clusterSecurityFile und -clusterSecurityUrl
  Diese Argumente geben die Datei `objectGridSecurity.xml` an, die die Sicherheitseigenschaften beschreibt, die für alle Server (einschließlich Katalogservern und Containerservern) gelten. Ein Beispiel für eine solche Eigenschaft ist die Authentifikatorkonfiguration, die die Benutzer-Registry und das Authentifizierungsverfahren darstellt.

  -serverProps
  Gibt die Servereigenschaftendatei an, die die serverspezifischen Sicherheitseigenschaften enthält. Der Dateiname für diese Eigenschaft kann im herkömmlichen Dateipfadformat angegeben werden, z. B. `c:/tmp/og/catalogserver.props`.

  Ein Beispiel zum Starten eines sicheren Katalogservice finden Sie in Schritt 2 des Lernprogramms zur Java-SE-Sicherheit in der *Produktübersicht*. Ein Beispiel für

die Datei objectGridSecurity.xml finden Sie im Abschnitt „XML-Sicherheitsdeskriptordatei” auf Seite 379.

Containerprozesse starten

Sie können eXtreme Scale über die Befehlszeile unter Verwendung einer Implementierungstopologie oder einer Datei server.properties starten.

Informationen zu diesem Vorgang

Zum Starten eines Containerprozesses benötigen Sie eine ObjectGrid-XML-Datei. Die ObjectGrid-XML-Datei gibt an, welche eXtreme-Scale-Server im Container enthalten sind. Stellen Sie sicher, dass Ihr Container jedes ObjectGrid in der XML aufnehmen kann, die Sie übergeben. Alle Klassen, die diese ObjectGrids voraussetzen, müssen im Klassenpfad des Containers enthalten sein. Weitere Informationen zur ObjectGrid-XML-Datei finden Sie im Abschnitt „Datei objectGrid.xsd” auf Seite 162.

Vorgehensweise

- **Starten Sie den Containerprozess über die Befehlszeile.**
  1. Navigieren Sie über die Befehlszeile zum Verzeichnis "bin":
     ```
     cd ObjectGrid-Stammverzeichnis/bin
     ```
  2. Führen Sie den folgenden Befehl aus:
     ```
     startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

     Wichtig: Im Container wird die Option **-catalogServiceEndPoints** verwendet, um auf den ORB-Host (Object Request Broker) und -Port des Katalogservice zu verweisen. Der Katalogservice verwendet die Optionen **-listenerHost** und **-listenerPort**, um den Host und den Port für die ORB-Bindung anzugeben, oder er akzeptiert die Standardbindung. Wenn Sie einen Container starten, verwenden Sie die Option **-catalogServiceEndPoints**, um die Werte zu referenzieren, die an die Optionen **-listenerHost** und **-listenerPort** im Katalogservice übergeben werden. Wenn die Optionen **-listenerHost** und **-listenerPort** beim Starten des Katalogservice nicht verwendet werden, stellt der ORB eine Bindung zu Port 2809 auf dem lokalen Host für den Katalogservice her. Verwenden Sie die Option **-catalogServiceEndPoints** nicht, um die Hosts und Ports zu referenzieren, die an die Option **-catalogServiceEndPoints** im Katalogservice übergeben wurden. Im Katalogservice wird die Option **-catalogServiceEndPoints** verwendet, um die erforderlichen Ports für die statische Serverkonfiguration anzugeben.
     Dieser Prozess wird mit c0 identifiziert, dem ersten Argument, das an das Script übergeben wird. Verwenden Sie die Datei companyGrid.xml, um den Container zu starten. Wenn Ihr Katalogserver-ORB auf einem anderen Host ausgeführt wird als Ihr Container oder wenn er einen vom Standard abweichenden Port verwendet, müssen Sie das Argument **-catalogServiceEndPoints** verwenden, um die Verbindung zum ORB herzustellen. Für dieses Beispiel wird angenommen, dass ein einzelner Katalogservice an Port 2809 auf MyServer1.company.com aktiv ist.
- **Starten Sie den Container über eine Implementierungsrichtlinie.**

  Obwohl dies nicht erforderlich ist, wird eine Implementierungsrichtlinie für den Containerstart empfohlen. Die Implementierungsrichtlinie wird verwendet, um die Partitionierung und Replikation für eXtreme Scale zu konfigurieren. Die Implementierungsrichtlinie kann auch verwendet werden, um das Verteilungsverhalten zu beeinflussen. Da im vorherigen Beispiel keine Implementierungsrichtliniendatei angegeben wurde, empfängt das Beispiel alle Standardwerte für

Replikation, Partitionierung und Verteilung. Deshalb sind die Maps im CompanyGrid in einem einzigen MapSet enthalten. Das MapSet wird nicht partitioniert oder repliziert. Weitere Informationen zu Implementierungsrichtliniendateien finden Sie im Abschnitt „XML-Deskriptordatei für Implementierungsrichtlinie" auf Seite 177. Im folgenden Beispiel wird die Datei `companyGridDpReplication.xml` verwendet, um eine Container-JVM, die JVM "c0", zu starten:

1. Navigieren Sie über die Befehlszeile zum Verzeichnis "bin":

   ```
   cd ObjectGrid-Stammverzeichnis/bin
   ```

2. Führen Sie den folgenden Befehl aus:

   ```
   startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml
   -deploymentPolicyFile ../xml/companyGridDpReplication.xml
   -catalogServiceEndPoints MyServer1.company.com:2809
   ```

Anmerkung: Wenn Sie Java-Klassen haben, die in einem bestimmten Verzeichnis gespeichert sind, können Sie den Server wie folgt mit Argumenten starten, anstatt das Script "StartOgServer" zu ändern: `-jvmArgs -cp C:\ . . . \DirectoryPOJOs\POJOs.jar`
In der Datei `companyGridDpReplication.xml` enthält ein einziges MapSet alle Maps. Dieses MapSet wird in 10 Partitionen aufgeteilt. Jede Partition hat ein synchrones Replikat und keine asynchronen Replikate. Jeder Container, der die Implementierungsrichtlinie `companyGridDpReplication.xml` in Kombination mit der ObjectGrid-XML-Datei `companyGrid.xml` verwendet, kann CompanyGrid-Shards aufnehmen. Starten Sie eine weitere Container-JVM, die JVM "c1":

1. Navigieren Sie über die Befehlszeile zum Verzeichnis "bin":

   ```
   cd ObjectGrid-Stammverzeichnis/bin
   ```

2. Führen Sie den folgenden Befehl aus:

   ```
   startOgServer.sh c1 -objectGridFile ../xml/companyGrid.xml
   -deploymentPolicyFile ../xml/companyGridDpReplication.xml
   -catalogServiceEndPoints MyServer1.company.com:2809
   ```

Jede Implementierungsrichtlinie enthält ein oder mehrere Elemente "objectgridDeployment". Wenn ein Container gestartet wird, veröffentlicht er seine Implementierungsrichtlinie im Katalogservice. Der Katalogservice untersucht jedes Element "objectgridDeployment". Wenn der Wert des Attributs "objectgridName" mit dem Wert des Attributs "objectgridName" eines zuvor empfangenen Elements "objectgridDeployment" übereinstimmt, wird das letzte Element "objectgridDeployment" ignoriert. Das erste Element "objectgridDeployment", das für ein bestimmtes Attribut "objectgridName" empfangen wird, wird als Master verwendet. Angenommen, die JVM "c2" verwendet eine Implementierungsrichtlinie, die das MapSet in eine andere Anzahl von Partitionen aufteilt:

companyGridDpReplicationModified.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy
  ../deploymentPolicy.xsd"
    xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">

     <objectgridDeployment objectgridName="CompanyGrid">
        <mapSet name="mapSet1" numberOfPartitions="5"
                minSyncReplicas="1" maxSyncReplicas="1"
            maxAsyncReplicas="0">
            <map ref="Customer" />
            <map ref="Item" />
            <map ref="OrderLine" />
            <map ref="Order" />
```

```
        </mapSet>
    </objectgridDeployment>

</deploymentPolicy>
```

Jetzt können Sie eine dritte JVM, die JVM "c2", starten.

1. Navigieren Sie über die Befehlszeile zum Verzeichnis "bin":
   ```
   cd ObjectGrid-Stammverzeichnis/bin
   ```
2. Führen Sie den folgenden Befehl aus:
   ```
   startOgServer.sh c2 -objectGridFile ../xml/companyGrid.xml
   -deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml
   -catalogServiceEndPoints MyServer1.company.com:2809
   ```

Der Container in der JVM "c2" wird mit einer Implementierungsrichtlinie gestartet, die 5 Partitionen für mapSet1 angibt. Der Katalogservice enthält jedoch bereits die Masterkopie des objectgridDeployment-Objekt für das CompanyGrid. Als die JVM "c0" gestartet wurde, waren 10 Partitionen für dieses MapSet vorhanden. Weil dies der erste Container war, der gestartet wurde und seine Implementierungsrichtlinie veröffentlicht hat, wurde seine Implementierungsrichtlinie als Master definiert. Deshalb wird jeder objectgridDeployment-Attributwert, der `CompanyGrid` entspricht, in einer nachfolgenden Implementierungsrichtlinie ignoriert.

- **Starten Sie einen Container über eine Servereigenschaftendatei.**

  Sie können eine Servereigenschaftendatei verwenden, um die Trace-Erstellung zu und die Sicherheit für einen Container zu konfigurieren. Führen Sie die folgenden Befehle aus, um Container "c3" über eine Servereigenschaftendatei zu starten:

  1. Navigieren Sie über die Befehlszeile zum Verzeichnis "bin":
     ```
     cd ObjectGrid-Stammverzeichnis/bin
     ```
  2. Führen Sie den folgenden Befehl aus:
     ```
     startOgServer.sh c3 -objectGridFile ../xml/companyGrid.xml
     -deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     -serverProps ../serverProps/server.properties
     ```

  Im Folgenden sehen Sie eine Beispieldatei `server.properties`:

  ```
  server.properties
  workingDirectory=
  traceSpec=*=all=disabled
  systemStreamToFileEnabled=trueenableMBeans=true
  memoryThresholdPercentage=50
  ```

  Dies ist die Basisservereigenschaftendatei, in der die Sicherheit nicht aktiviert ist. Weitere Informationen zur Datei `server.properties` finden Sie im Abschnitt „Servereigenschaftendatei" auf Seite 188.

Script "startOgServer"

Das Script "startOgServer" startet Server. Sie können beim Starten Ihrer Server eine Vielzahl von Parametern verwenden, um die Trace-Erstellung zu aktivieren, Portnummern anzugeben usw.

Zweck

Sie können das Script "startOgServer" verwenden, um Server zu starten.

Position

Sie finden das Script "startOgServer" im Verzeichnis "bin" des Stammverzeichnisses, z. B.:

```
cd ObjectGrid-Stammverzeichnis/bin
```

Anmerkung: Wenn Sie Java-Klassen haben, die in einem bestimmten Verzeichnis gespeichert sind, können Sie den Server wie folgt mit Argumenten starten, anstatt das Script "startOgServer" zu ändern: `-jvmArgs -cp C:\ . . . \DirectoryPOJOs\POJOs.jar`

Syntax für Katalogserver

Verwenden Sie zum Starten eines Katalogservers die folgenden Befehle:

Windows

```
startOgServer.bat <Server> [Optionen]
```

UNIX

```
startOgServer.sh <Server>[Optionen]
```

Verwenden Sie zum Starten des konfigurierten Standardkatalogservers die folgenden Befehle:

Windows

```
startOgServer.bat catalogServer
```

UNIX

```
startOgServer.sh catalogServer
```

Optionen für das Starten von Katalogservern

Parameter für das Starten eines Katalogservers:

-catalogServiceEndPoints <Server:Serverhost:Clientport:Peer-Port,Server:Serverhost:Clientport:Peer-Port>
Im Container wird die Option **-catalogServiceEndpoints** verwendet, um auf den ORB-Host (Object Request Broker) und -Port des Katalogservice zu verweisen.

-clusterSecurityFile <XML-Sicherheitsdatei des Clusters>
Gibt die Position der XML-Sicherheitsdeskriptordatei an.

-clusterSecurityUrl <URL der Sicherheits-XML des Clusters>

-domain <Domänenname>

-listenerHost <Hostname>
Standardeinstellung: `localhost`

Gibt den Listener-Host für die Kommunikation mit Internet Inter-ORB Protocol (IIOP) an.

-listenerPort <Port>
Standardeinstellung: `2809`

Gibt den Listener-Port für die Kommunikation mit IIOP an.

-haManagerPort <Port>
Standardeinstellung: Der Name des Peer-Ports. Wenn Sie diese Eigenschaft nicht definieren, generiert der Katalogservice automatisch einen verfügbaren Port.

Gibt die Nummer des vom High Availability Manager verwendeten Ports an.

-serverProps <Servereigenschaftendatei>
Gibt die Servereigenschaftendatei an, die die serverspezifischen Sicherheitseigenschaften enthält. Der Dateiname für diese Eigenschaft kann im herkömmlichen Dateipfadformat angegeben werden, z. B. `c:/tmp/og/catalogserver.props`.

-JMXServicePort <Port>
Standardeinstellung: `1099`

Gibt die Portnummer für die Kommunikation mit Java Management Extensions (JMX) an.

-traceSpec <Trace-Spezifikation>
Gibt eine Zeichenfolge an, die den Umfang des Trace angibt, der beim Serverstart aktiviert wird.

Beispiel:

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <Trace-Datei>
Gibt den Pfad der Datei an, in der Trace-Informationen gespeichert werden.

Beispiel: `../logs/c4Trace.log`

-timeout <Sekunden>
Gibt das Zeitlimit (in Sekunden) für den Serverstart an.

-script <Scriptdatei>
Erstellt ein angepasstes Script für Befehle, die Sie angeben, um Katalogserver oder -container zu starten und dann wie gewünscht zu parametrisieren oder zu bearbeiten.

-jvmArgs <JVM-Argumente>
Gibt eine Gruppe von JVM-Argumenten an. Jeder Parameter hinter dem Parameter **-jvmArgs** wird verwendet, um die Server-JVM zu starten. Wenn der Parameter **-jvmArgs** verwendet wird, müssen Sie sicherstellen, dass er das letzte optionale Scriptargument ist, das angegeben wird.

Beispiel:`-jvmArgs -Xms256M -Xmx1G`

Syntax für Containerserver Windows

```
startOgServer.bat <Server> -objectgridFile <XML-Datei>
-deploymentPolicyFile <XML-Datei> [Optionen]
```

Windows

```
startOgServer.bat <Server> -objectgridUrl <XML-URL>
-deploymentPolicyUrl <XML-URL> [Optionen]
```

UNIX

```
startOgServer.sh <Server> -objectgridFile <XML-Datei>
-deploymentPolicyFile <XML-Datei> [Optionen]
```

UNIX

```
startOgServer.sh <Server> -objectgridUrl <XML-URL>
-deploymentPolicyUrl <XML-URL> [Optionen]
```

Optionen für Containerserver

-catalogServiceEndPoints<Hostname:Port,Hostname:Port>
Standardeinstellung: localhost:2809

-deploymentPolicyFile <XML-Datei_für_Implementierungsrichtlinien>
Gibt den Pfad der Datei für die Implementierungsrichtlinien an.

Beispiel: `../xml/SimpleDP.xml`

-deploymentPolicyUrl <URL_der_Implementierungsrichtlinie>

-objectgridFile <ObjectGrid-XML-Deskriptordatei>
Gibt den Pfad der ObjectGrid-Deskriptordatei an.

-objectgridUrl <URL_des_ObjectGrid-Deskriptors>

-listenerHost <Hostname>
Standardeinstellung: `localhost`

Gibt den Listener-Host für die Kommunikation mit Internet Inter-ORB Protocol (IIOP) an.

-listenerPort <Port>
Standardeinstellung: `2809`

Gibt den Listener-Port für die Kommunikation mit IIOP an.

-serverProps <Servereigenschaftendatei>
Gibt den Pfad der Servereigenschaftendatei an.

Beispiel:`../security/server.props`

-zone <Zonenname>
Gibt die für alle Container im Server zu verwendende Zone an.

-traceSpec <Trace-Spezifikation>
Gibt eine Zeichenfolge an, die den Umfang des Trace angibt, der beim Serverstart aktiviert wird.

Beispiel:

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <Trace-Datei>
Gibt den Pfad der Datei an, in der Trace-Informationen gespeichert werden.

Beispiel: `../logs/c4Trace.log`

-timeout <Sekunden>
Gibt das Zeitlimit (in Sekunden) für den Serverstart an.

-script <Scriptdatei>
Erstellt ein angepasstes Script für Befehle, die Sie angeben, um Katalogserver oder -container zu starten und dann wie gewünscht zu parametrisieren oder zu bearbeiten.

-jvmArgs <JVM-Argumente>
Gibt eine Gruppe von JVM-Argumenten an. Jeder Parameter hinter dem Parameter **-jvmArgs** wird verwendet, um die Server-JVM zu starten. Wenn der Pa-

rameter **-jvmArgs** verwendet wird, müssen Sie sicherstellen, dass er das letzte optionale Scriptargument ist, das angegeben wird.

Beispiel:-jvmArgs -Xms256M -Xmx1G

Eigenständige eXtreme-Scale-Server stoppen

Sie können das Script stopOgServer verwenden, um Serverprozesse zu stoppen.

Vorgehensweise

- Stoppen Sie die eXtreme-Scale-Containerprozesse.
  1. Navigieren Sie über die Befehlszeile zum Verzeichnis "bin".
     cd *ObjectGrid-Stammverzeichnis*/bin
  2. Führen Sie das Script stopOgServer aus, um den Server zu stoppen. Im folgenden Beispiel wird der Server "c0" gestoppt:
     stopOgServer *c0* -catalogServiceEndPoints *MyServer1.company.com:2809*

     Verwenden Sie dasselbe Script, um mehrere Server wie folgt mit einer durch Kommas begrenzten Liste zu stoppen:
     stopOgServer c0,c1,c2 -catalogServiceEndPoints MyServer1.company.com:2809

     Achtung: Die Option **-catalogServiceEndPoints** muss mit der Option **-catalogServiceEndPoints** übereinstimmen, die zum Starten des Containers verwendet wird. Wenn die Option **-catalogServiceEndPoints** nicht zum Starten des Containers verwendet wird, werden wahrscheinlich die Standardewerte "localhost" oder der Hostname und ORB-Port 2809 verwendet, um eine Verbindung zum Katalogservice herzustellen. Andernfalls verwenden Sie die an **-listenerHost** und **-listenerPort** im Katalogservice übergebenen Werte. Wenn die Optionen **-listenerHost** und **-listenerPort** beim Starten des Katalogservice nicht verwendet werden, stellt der ORB eine Bindung zu Port 2809 auf dem lokalen Host für den Katalogservice her.
- Stoppen Sie die eXtreme-Scale-Katalogserviceprozesse.
  1. Navigieren Sie über die Befehlszeile zum Verzeichnis "bin".
     cd ObjectGrid-Stammverzeichnis/bin
  2. Führen Sie das Script stopOgServer aus, um den Server zu stoppen.
     stopOgServer.sh *Katalogserver* -catalogServiceEndPoints *MyServer1.company.com:2809*

  Achtung: Wenn Sie einen Katalogservice stoppen, wird die Option **-catalogServiceEndPoints** verwendet, um auf den ORB-Host und den ORB-Port im Katalogservice zu verweisen. Der Katalogservice verwendet die Optionen **-listenerHost** und **-listenerPort**, um den Host und den Port für die ORB-Bindung anzugeben, oder er akzeptiert die Standardbindung. Wenn die Optionen **-listenerHost** und **-listenerPort** beim Starten des Katalogservice nicht verwendet werden, stellt der ORB eine Bindung zu Port 2809 auf dem lokalen Host für den Katalogservice her. Die Option **-catalogServiceEndPoints** wird beim Stoppen eines Katalogservice anders verwendet als beim Starten eines Katalogservice.

  Für das Starten eines Katalogservice sind Peer-Zugriffsports und Clientzugriffsports erforderlich, wenn die Standardports nicht verwendet werden. Für das Stoppen eines Katalogservice wird nur der ORB-Port benötigt.
- Aktivieren Sie die Trace-Erstellung für den Prozess zum Stoppen des Servers. Wenn ein Container nicht gestoppt werden kann, können Sie die Trace-Erstellung als Unterstützung für die Fehlerbehebung aktivieren. Zum Aktivieren der Trace-Erstellung beim Stoppen eines Servers fügen Sie den Stoppbefehlen die Parameter **-traceSpec** und **-traceFile** hinzu. Der Parameter **-traceSpec** gibt den Typ

des zu aktivierenden Trace an, und der Parameter **-traceFile** gibt den Pfad und den Namen der für die Trace-Daten zu erstellenden und zu verwendenden Datei an.

1. Navigieren Sie über die Befehlszeile zum Verzeichnis "bin".
   ```
   cd ObjectGrid-Stammverzeichnis/bin
   ```
2. Führen Sie das Script stopOgServer mit aktivierter Trace-Erstellung aus.
   ```
   stopOgServer.sh c4 -catalogServiceEndPoints MyServer1.company.com:2809
    -traceFile ../logs/c4Trace.log -traceSpec ObjectGrid=all=enabled
   ```

   Suchen Sie nach der Trace-Erstellung nach Fehlern, die sich auf Portkonflikte, fehlende Klassen, fehlende oder ungültige XML-Dateien oder Stack-Traces beziehen. Empfohlene Trace-Spezifikation für den Start sind:

   – `ObjectGrid=all=enabled`

   – `ObjectGrid*=all=enabled`

   Informationen zu allen Optionen für die Trace-Spezifikation finden Sie im Abschnitt „Trace-Optionen” auf Seite 457.

Script "stopOgServer"

Das Script "stopOgServer" stoppt Server.

Zweck

Sie können das Script "stopOgServer" zum Stoppen eines Servers verwenden, indem Sie den Namen des Servers und die Endpunkte des Katalogservice angeben.

Position

Sie finden das Script "stopOgServer" im Verzeichnis "bin" des Stammverzeichnisses, z. B.:

```
cd ObjectGrid-Stammverzeichnis/bin
```

Syntax

Verwenden Sie zum Stoppen eines Katalogservers die folgenden Befehle:

Windows

```
stopOgServer.bat <Server> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [Optionen]
```

UNIX

```
stopOgServer.sh <server> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [Optionen]
```

Verwenden Sie zum Stoppen eines ObjectGrid-Containerservers die folgenden Befehle:

Windows

```
stopOgServer.bat <Server> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [Optionen]
```

UNIX

```
startOgServer.sh -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [Optionen]
```

Optionen

-clientSecurityFile <Sicherheitseigenschaftendatei>

-traceSpec <Trace-Spezifikation>
Gibt eine Zeichenfolge an, die den Umfang des Trace angibt, der beim Serverstart aktiviert wird.

Beispiel:

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <Trace-Datei>
Gibt den Pfad der Datei an, in der Trace-Informationen gespeichert werden.

Beispiel: `../logs/c4Trace.log`

-jvmArgs <JVM-Argumente>
Jeder Parameter hinter der Option "-jvmArgs" wird verwendet, um die Server-JVM zu starten. Wenn die Option "-jvmArgs" verwendet wird, müssen Sie sicherstellen, dass er das letzte optionale Scriptargument ist, das angegeben wird.

Sichere eXtreme-Scale-Server starten und stoppen

Server müssen für die Implementierungsumgebung häufig sicher sein, und dies erfordert eine spezielle Konfiguration für das Starten und Stoppen der Server.

Sicheren Server in einer Java-SE-Umgebung starten

Sie können einen Katalogservice oder Containerserver wie folgt starten.

Sicheren eXtreme-Scale-Katalogservice starten

Für das Starten eines sicheren eXtreme-Scale-Katalogserviceprozesses sind zwei weitere Sicherheitskonfigurationsdateien erforderlich:

XML-Sicherheitsdeskriptordatei: Die XML-Sicherheitsdeskriptordatei beschreibt die Sicherheitseigenschaften, die für alle Server (einschließlich Katalogservern und Containerservern) gelten. Ein Beispiel für eine solche Eigenschaft ist die Authentifikatorkonfiguration, die die Benutzer-Registry und das Authentifizierungsverfahren darstellt.

Servereigenschaftendatei. Die Servereigenschaftendatei konfiguriert die für den Server spezifischen Sicherheitseigenschaften.

Wenn Sie den Befehl "startOgServer.sh" bzw. "startOgServer.cat" zum Starten eines sicheren eXtreme-Scale-Katalogserviceprozesses verwenden, können Sie "-clusterSecurityFile" oder "-clusterSecurityUrl" verwenden, um die XML-Sicherheitsdeskriptordatei als Dateityp oder URL-Typ festzulegen, und Sie können "–serverProps" verwenden, um die Servereigenschaftendatei zu setzen.

Sicheren eXtreme-Scale-Containerserver starten

Für das Starten eines sicheren eXtreme-Scale-Containerservers ist eine einzige Sicherheitskonfigurationsdatei erforderlich:

- **Servereigenschaftendatei:** Die Servereigenschaftendatei konfiguriert die für den Server spezifischen Sicherheitseigenschaften. Weitere Einzelheiten finden Sie im Abschnitt „Servereigenschaftendatei” auf Seite 188.

Wenn Sie den Befehl "startOgServer.sh" bzw. "startOgServer.cat" zum Starten eines sicheren eXtreme-Scale-Containerservers verwenden, können Sie mit "–serverProps" die Servereigenschaftendatei festlegen. Es gibt weitere Methoden zum Festlegen der Servereigenschaftendatei. Weitere Informationen hierzu finden Sie in der Beschreibung der Servereigenschaftendatei.

Einzelheiten zur Verwendung des Befehls "startOgServer.sh" bzw. "startOgServer-.bat" und der zugehörigen Optionen finden Sie im Abschnitt „Script "startOgServer"” auf Seite 340.

Sicheren eXtreme-Scale-Server stoppen

Für das Stoppen eines sicheren eXtreme-Scale-Katalogserviceprozesses bzw. Containerservers ist eine einzige Sicherheitskonfigurationsdatei erforderlich:

- **Clienteigenschaftendatei:** Die Clienteigenschaftendatei kann zum Konfigurieren der Clientsicherheitseigenschaften verwendet werden. Die Clientsicherheitseigenschaften sind erforderlich, damit ein Client eine Verbindung zu einem sicheren Server herstellen kann. Weitere Einzelheiten finden Sie im Abschnitt „Clienteigenschaftendatei” auf Seite 207.

Wenn Sie den Befehl "stopOgServer.sh" bzw. "stopOgServer.cat" zum Stoppen eines sicheren eXtreme-Scale-Katalogserviceprozesses oder -Containerservers verwenden, können Sie mit "-clientSecurityFile" die Sicherheitseigenschaften des Clients festlegen.

Einzelheiten zur Verwendung des Befehls "stopOgServer.sh" bzw. "stopOgServer-.cat" und der zugehörigen Optionen finden Sie im Abschnitt „Script "stopOgServer"” auf Seite 345.

Sicheren Server in WebSphere Application Server starten

Das Starten eines sicheren ObjectGrid-Servers in WebSphere Application Server verläuft ähnlich wie das Starten eines nicht sicheren ObjectGrid-Servers, abgesehen davon, dass Sie die Sicherheitskonfigurationsdateien übergeben müssen. Anstatt wie in der J2SE-Umgebung das Argument "-[EIGENSCHAFTENDATEI]" (z. B. -serverProps) im Befehl zu übergeben, verwenden Sie das Argument "-D[EIGENSCHAFTENDATEI] in den generischen JVM-Argumenten.

Sicheren Katalogservice in WebSphere Application Server starten

Ein Katalogserver enthält zwei verschiedene Stufen von Sicherheitsinformationen:

- `-Dobjectgrid.cluster.security.xml.url`: Gibt die Datei "objectGridSecurity.xml" an, die die Sicherheitseigenschaften beschreibt, die für alle Server (einschließlich Katalogservern und Containerservern) gelten. Ein Beispiel ist die Authentifikatorkonfiguration, die die Benutzer-Registry und das Authentifizierungsverfahren darstellt. Der Dateiname für diese Eigenschaft muss im URL-Format angegeben werden, z. B. "file:///tmp/og/objectGridSecurity.xml".
- `-Dobjectgrid.server.props`: Gibt die Servereigenschaftendatei an, die die serverspezifischen Sicherheitseigenschaften enthält. Der für diese Eigenschaft angegebene Dateiname kann im herkömmlichen Dateipfadformat angegeben werden, z.

B. "c:/tmp/og/catalogserver.props". Die Verwendung von "-Dobjectgrid.security-.server.props" ist veraltet, aber Sie können diese Option für die Abwärtskompatibilität weiterhin verwenden.

Zum Starten eines sicheren Katalogservice in WebSphere Application Server folgen Sie den Anweisungen unter "Integriert in WebSphere Application Server" im Abschnitt „Grid-Sicherheit” auf Seite 362.

Als Nächstes definieren Sie die Sicherheitseigenschaft in den generischen JVM-Argumenten des Prozesses:

```
-Dobjectgrid.cluster.security.xml.url=file:///tmp/og/
objectGridSecurity.xml-Dobjectgrid.server.props=/tmp/og/
catalog.server.props
```

Im Folgenden wird beschrieben, wie Sie die Eigenschaft den generischen JVM-Argumenten hinzufügen:

- Erweitern Sie den Eintrag "Systemverwaltung" in der Task-Ansicht auf der linken Seite.
- Klicken Sie auf den Prozess von WebSphere Application Server, in dem der Katalogservice implementiert ist, z. B. "Deployment Manager".
- Erweitern Sie auf der rechten Seite den Eintrag "Java- und Prozessverwaltung" unter "Serverinfrastruktur".
- Klicken Sie auf "Prozessdefinition".
- Klicken Sie auf "Java Virtual Machine" unter "Weitere Eigenschaften".
- Geben Sie die Eigenschaften im Textfeld "Generische JVM-Argumente" ein.

Sicheren Containerserver in WebSphere Application Server starten

Ein Containerserver erhält bei der Verbindungsherstellung zum Katalogserver alle Sicherheitskonfigurationen, die in der Datei "objectGridSecurity.xml" konfiguriert sind, wie z. B. die Authentifikatorkonfiguration oder das Zeitlimit für Anmeldesitzungen. Außerdem muss ein Containerserver seine eigenen serverspezifischen Sicherheitseigenschaften in der Eigenschaft "-Dobjectgrid.server.props" konfigurieren.

Die Eigenschaft "-Dobjectgrid.server.props" wird an Stelle der Eigenschaft "-Dobjectgrid.security.server.props" verwendet, weil in diese Eigenschaftendatei auch andere nicht sicherheitsrelevante Eigenschaften aufgenommen werden. Der Dateiname für diese Eigenschaft kann im herkömmlichen Dateipfadformat angegeben werden, z. B. `c:/tmp/og/server.props`.

Führen Sie dieselben Schritte wie zuvor aus, um die Sicherheitseigenschaft den generischen JVM-Argumenten hinzuzufügen.

WebSphere eXtreme Scale mit WebSphere Application Server verwalten

Sie können Katalogservice- und Containerserverprozesse in WebSphere Application Server ausführen. Der Prozess zum Konfigurieren dieser Server unterscheidet sich von dem in einer eigenständigen Konfiguration. Der Katalogservice kann automatisch in Servern oder im Deployment Manager von WebSphere Application Server gestartet werden. Der Containerprozess wird gestartet, wenn eine eXtreme-Scale-Anwendung in der Umgebung von WebSphere Application Server implementiert und gestartet wird.

Katalogserviceprozess in einer Umgebung mit WebSphere Application Server starten

eXtreme-Scale-Katalogserver kann automatisch in einer Umgebung von WebSphere Application Server oder WebSphere Application Server Network Deployment gestartet werden. Sie können den Katalogservice so konfigurieren, dass er in einem beliebigen Prozess in der WebSphere-Zelle ausgeführt werden kann. Ein Katalogservice mit einem einzigen Server ohne Cluster ist für Entwicklungsumgebungen akzeptabel. Für Produktionsumgebungen müssen Sie eine Katalogservicedomäne mit mehreren Katalogservern verwenden.

Vorbereitende Schritte

- Sie müssen WebSphere Application Server und WebSphere eXtreme Scale installieren.

  Wenn Sie eine grafisch unterstützte Installation von eXtreme Scale durchführen, haben Sie die Möglichkeit, vorhandene Profile zu erweitern, einschließlich des Deployment-Manager-Profils. Sie können Profile auch nach der Installation mit Profile Management Tool (PMT) erweitern, entweder mit der GUI-Version oder über die Befehlszeile (manageprofiles.sh | bat). Nach der Installation des Produkts erstellte Profile können im Rahmen des Profilerstellungsprozesses oder später mit PMT erweitert werden. Es gibt keine PMT-GUI für 64-Bit-Installationen von WebSphere Application Server. In diesen Fällen verwenden Sie das Script "manageprofiles" über die Befehlszeile.

  Weitere Informationen finden Sie unter Kapitel 3, „WebSphere eXtreme Scale installieren und implementieren”, auf Seite 17.

- 7.1+ Definieren Sie eine Katalogservicedomäne. Weitere Informationen finden Sie unter „Katalogservicedomänen in WebSphere Application Server erstellen” auf Seite 350.

Informationen zu diesem Vorgang

Der Katalogserviceprozess kann in einem Prozess von WebSphere Application Server ausgeführt werden. Wo und wie der Katalogservice ausgeführt wird, richtet sich nach der verwendeten Edition von WebSphere Application Server:

WebSphere Application Server (Basis)
: Der Katalogservice wird im Anwendungsserverprozess ausgeführt. Standardmäßig ist der automatische Start des Katalogservice *nicht* aktiviert.

WebSphere Application Server Network Deployment
: Der Katalogservice wird im Deployment-Manager-Prozess automatisch ausgeführt, aber Sie können ihn für die Ausführung in einem oder mehreren Node-Agent- oder Anwendungsserverprozessen konfigurieren.

Veraltetes Feature:  In früheren Releases haben Sie die angepasste Eigenschaft "catalog.services.cluster" definiert, um eine Gruppe von Katalogservern in Ihrer Umgebung von WebSphere Application Server zu definieren. Haben Sie diese angepasste Eigenschaft mit einem früheren Release konfiguriert, sind die Einstellungen immer noch gültig. Wenn Sie jedoch eine neue Konfiguration erstellen, können Sie dieselbe Konfiguration durch Erstellen einer Katalogservicedomäne über die Administrationskonsole von WebSphere Application Server erreichen. Weitere Informationen finden Sie unter „Katalogservicedomänen in WebSphere Application Server erstellen” auf Seite 350.

Einschränkung: Server desselben Namens in der Umgebung von WebSphere Application Server sind nicht zulässig, wenn der Object Request Broker (ORB) für die Kommunikation verwendet wird. Sie können diese Einschränkung aufheben, indem Sie die Systemeigenschaft
-Dcom.ibm.websphere.orb.uniqueServerName=true für die Prozesse angeben, die denselben Namen haben. Im Folgenden finden Sie verschiedene Beispiele: Es werden Server mit dem Namen "server1" auf jedem Knoten als Grid von Katalogservices verwendet, oder es werden mehrere Node Agents verwendet, um ein Grid von Katalogservices zu bilden.

Vorgehensweise

- **Katalogserver im Basisprodukt WebSphere Application Server** starten:

  Wenn WebSphere eXtreme Scale in ein nicht eingebundenes Profil von WebSphere Application Server integriert wird, wird der Katalogservice mit Ausnahme der folgenden beiden Fälle nicht automatisch gestartet:

  – Eine Anwendung, die für den automatischen Start eines eXtreme-Scale-Containers konfiguriert ist: Katalogservice und -container werden automatisch gestartet. Dieses Feature vereinfacht die Komponententests in Entwicklungsumgebungen wie Rational Application Developer, weil Sie einen Katalogservice nicht explizit starten müssen. Weitere Informationen finden Sie unter „eXtreme-Scale-Container automatisch in Anwendungen von WebSphere Application Server starten” auf Seite 357.

  – Es ist eine Katalogservicedomäne definiert.

- **Katalogservice in WebSphere Application Server Network Deployment** starten: Der Katalogservice wird in den folgenden Szenarien in einer Zelle von WebSphere Application Server Network Deployment gestartet:

  – Wenn WebSphere eXtreme Scale in WebSphere Application Server Network Deployment installiert ist, wird der Katalogservice automatisch im Deployment-Manager-Prozess gestartet, wenn er erweitert wurde, um eine schnelle Entwicklung ohne Vorbereitungs- oder Anpassungsaufwand zuzulassen.

  – Sie definieren eine Katalogservicedomäne. Die Katalogservicedomäne ist eine Sammlung definierter Katalogserver. Diese Katalogserver können in vorhandenen Anwendungsservern in der Umgebung von WebSphere Application Server Network Deployment gestartet werden, oder Sie können ferne Server definieren. Weitere Informationen finden Sie unter „Katalogservicedomänen in WebSphere Application Server erstellen”.

  Achtung: Fassen Sie die Katalogservices nicht durch Kollokation mit Containerservern von eXtreme Scale in einer Produktionsumgebung zusammen. Führen Sie den Katalogservice in mehreren Node-Agent-Prozessen oder in einem Anwendungsserver ohne eine eXtreme-Scale-Anwendung aus.

  Nach der Definition einer Katalogservicedomäne müssen Sie jeden identifizierten Server erneut starten. Wenn Sie diese Server erneut starten, wird die Katalogservicedomäne automatisch gestartet.

Katalogservicedomänen in WebSphere Application Server erstellen

Katalogservicedomänen definieren eine Gruppe von Katalogservern, die die Verteilung von Shards verwalten und die Vitalität (ordnungsgemäßer Betrieb) der Containerserver in Ihrem Daten-Grid überwachen.

Vorbereitende Schritte

- WebSphere eXtreme Scale in WebSphere Application Server installieren. Weitere Informationen finden Sie unter „WebSphere eXtreme Scale oder WebSphere eXtreme Scale Client mit WebSphere Application Server integrieren” auf Seite 23.

Informationen zu diesem Vorgang

Durch die Erstellung einer Katalogservicedomäne definieren Sie eine hoch verfügbare Sammlung von Katalogservern.

Diese Katalogserver können in WebSphere Application Server in einer einzigen Zelle oder in einer Stammgruppe ausgeführt werden. Die Katalogservicedomäne kann auch eine ferne Gruppe von Servern definieren, die in verschiedenen Java-SE-Prozessoren oder in anderen Zellen von WebSphere Application Server ausgeführt werden. Wenn Sie eine Katalogservicedomäne definieren, die Katalogserver in den Anwendungsservern innerhalb der Zelle platziert, werden die Stammgruppenmechanismen von WebSphere Application Server verwendet. Deshalb können die Member einer Katalogservicedomäne nicht die Grenzen einer Stammgruppe überschreiten, und deshalb kann sich eine Katalogservicedomäne nicht über mehrere Zellen erstrecken. WebSphere eXtreme Scale kann jedoch mehrere Zellen umspannen, indem Verbindungen zu einem Katalogserver über Zellengrenzen hinweg hergestellt werden, z. B. als eigenständige Katalogservicedomäne oder Katalogservicedomäne, die in eine andere Zelle integriert ist.

Achtung: Fassen Sie die Katalogservices nicht durch Kollokation mit Containerservern von WebSphere eXtreme Scale in einer Produktionsumgebung zusammen. Schließen Sie den Katalogservice in mehrere Node-Agent-Prozess oder in einen Anwendungsserver ein, der keine Anwendung von WebSphere eXtreme Scale enthält.

Sie können eine Katalogservicedomäne auch erstellen, wenn Sie im eigenständigen Modus arbeiten. Weitere Informationen finden Sie unter „Katalogservice in einer eigenständigen Umgebung starten” auf Seite 335.

Vorgehensweise

1. Erstellen Sie die Katalogservicedomäne.
   a. Klicken Sie in der Administrationskonsole von WebSphere Application Server auf **Systemverwaltung** → **WebSphere eXtreme Scale** → **Katalogservicedomänen** → **Neu**.
   b. Definieren Sie einen Namen, einen Standardwert und JMX-Authentifizierungsnachweise für Ihre Katalogservicedomäne.
   c. Fügen Sie Katalogserviceendpunkte hinzu. Sie können vorhandene Anwendungsserver auswählen oder ferne Server hinzufügen, in einen ein Katalogservice ausgeführt wird.
2. Testen Sie die Verbindung zu Ihrer Katalogservicedomäne.
   a. Klicken Sie in der Administrationskonsole von WebSphere Application Server auf **Systemverwaltung** → **WebSphere eXtreme Scale** → **Katalogservicedomänen**.
   b. Wählen Sie die Katalogservicedomäne aus, die Sie testen möchten, und klicken Sie anschließend auf **Verbindung testen**. Wenn Sie auf diese Schaltfläche klicken, werden alle definierten Endpunkte der Katalogservicedomäne nacheinander abgefragt. Wenn ein Endpunkt verfügbar ist, wird eine Nachricht zurückgegeben, in der gemeldet wird, dass die Verbindung zur Katalogservicedomäne erfolgreich hergestellt werden konnte.

3. Wenn Sie Katalogserver in vorhandenen Anwendungsservern erstellen, starten Sie die ausgewählten Server erneut. Der Katalogservice wird in den ausgewählten Servern automatisch gestartet.

Nächste Schritte

Sie können diese Konfiguration auch mit dem Tool "wsadmin" erstellen. Weitere Informationen zu den Befehlen finden Sie unter „Verwaltungs-Tasks für Katalogservicedomäne”.

Verwaltungs-Tasks für Katalogservicedomäne

Sie können die Scripting-Sprachen Jacl und Jython verwenden, um Katalogservicedomänen in Ihrer Konfiguration von WebSphere Application Server zu verwalten.

Voraussetzungen

WebSphere eXtreme Scale Client muss in der Umgebung von WebSphere Application Server installiert sein.

Alle Verwaltungs-Tasks auflisten

Führen Sie den folgenden Befehl mit wsadmin aus, um eine Liste aller Verwaltungs-Tasks für Katalogservicedomänen abzurufen.

```
wsadmin>$AdminTask help XSDomainManagement
```

Befehle

Die Verwaltungs-Tasks für Katalogservicedomänen umfassen die folgenden Befehle:

- „createXSDomain”
- „deleteXSDomain” auf Seite 353
- „getDefaultXSDomain” auf Seite 354
- „listXSDomains” auf Seite 354
- „modifyXSDomain” auf Seite 355
- „testXSDomainConnection” auf Seite 356
- „testXSServerConnection” auf Seite 356

createXSDomain

Der Befehl "createXSDomain" registriert eine neue Katalogservicedomäne.

Tabelle 19. Argumente für den Befehl "createXSDomain"

Argument	Beschreibung
-name (erforderlich)	Gibt den Namen der Katalogservicedomäne an, die Sie bearbeiten möchten.
-default	Gibt an, ob die Katalogservicedomäne die Standarddomäne für die Zelle ist. Der Standardwert ist `true`. (Boolescher Wert: `true` oder `false`)
-userID	Gibt die Benutzer-ID für die Katalogservicedomäne an.
-password	Gibt das Kennwort für die Katalogservicedomäne an.

Tabelle 19. Argumente für den Befehl "createXSDomain" (Forts.)

Argument	Beschreibung
-properties	Gibt angepasste Eigenschaften für die Katalogservicedomäne.

Tabelle 20. Argumente für den Schritt "defineDomainServers"

Argument	Beschreibung
-name	Gibt den Namen des Katalogserviceendpunkts an.
-hostName	Gibt den Namen des Hosts an, auf dem sich der Endpunkt befindet.
-endPoints	Gibt die Portnummern für den Endpunkt der Katalogservicedomäne an.
-properties	Gibt angepasste Eigenschaften für den Endpunkt der Katalogservicedomäne an.

Rückgabewert:

Beispielsyntax für den Stapelmodus

- Mit Jacl:

```
$AdminTask createXSDomain {-name TestDomain -default true -userID xsuser
-password ***** -defineDomainServers {{cs1 xhost1.ibm.com "" 5501,2809,1099}
{cs2 xhost2.ibm.com "" 5501,2809,1099}}}
```

- Mit Jython (Zeichenfolge):

```
AdminTask.createXSDomain('[-name TestDomain -default true -userID xsuser
-password ***** -defineDomainServers [[cs1 xhost1.ibm.com "" 5501,2809,1099]
[cs2 xhost2.ibm.com "" 5501,2809,1099]]]')
```

Beispielsyntax für den Dialogmodus:

- Mit Jacl:

```
$AdminTask createXSDomain {-interactive}
```

- Mit Jython (Zeichenfolge):

```
AdminTask.createXSDomain ('[-interactive]')
```

deleteXSDomain

Der Befehl "deleteXSDomain" löscht eine Katalogservicedomäne.

Erforderliche Parameter:

-name
Gibt den Namen der zu löschenden Katalogservicedomäne an.

Rückgabewert:

Beispielsyntax für den Stapelmodus

- Mit Jacl:

```
$AdminTask deleteXSDomain {-name TestDomain }
```

- Mit Jython (Zeichenfolge):

```
AdminTask.deleteXSDomain('[-name TestDomain ]')
```

Beispielsyntax für den Dialogmodus:

- Mit Jacl:

```
$AdminTask deleteXSDomain {-interactive}
```

- Mit Jython (Zeichenfolge):

```
AdminTask.deleteXSDomain ('[-interactive]')
```

getDefaultXSDomain

Der Befehl "getDefaultXSDomain" gibt die Standardkatalogservicedomäne für die Zelle zurück.

Erforderliche Parameter: Ohne

Rückgabewert: Der Name der Standardkatalogservicedomäne.

Beispielsyntax für den Stapelmodus

- Mit Jacl:

```
$AdminTask getDefaultXSDomain
```

- Mit Jython (Zeichenfolge):

```
AdminTask.getDefaultXSDomain
```

Beispielsyntax für den Dialogmodus:

- Mit Jacl:

```
$AdminTask getDefaultXSDomain {-interactive}
```

- Mit Jython (Zeichenfolge):

```
AdminTask.getDefaultXSDomain ('[-interactive]')
```

listXSDomains

Der Befehl "listXSDomains" gibt eine Liste vorhandener Katalogservicedomänen zurück.

Erforderliche Parameter: Ohne

Rückgabewert: Eine Liste aller Katalogservicedomänen in der Zelle.

Beispielsyntax für den Stapelmodus

- Mit Jacl:

```
$AdminTask listXSDomains
```

- Mit Jython (Zeichenfolge):

```
AdminTask.listXSDomains
```

Beispielsyntax für den Dialogmodus:

- Mit Jacl:

```
$AdminTask listXSDomains {-interactive}
```

- Mit Jython (Zeichenfolge):

```
AdminTask.listXSDomains ('[-interactive]')
```

modifyXSDomain

Der Befehl "modifyXSDomain" ändert eine vorhandene Katalogservicedomäne.

Tabelle 21. Argumente für den Befehl "modifyXSDomain"

Argument	Beschreibung
-name (erforderlich)	Gibt den Namen der Katalogservicedomäne an, die Sie bearbeiten möchten.
-default	Wenn dieses Argument auf `true` gesetzt wird, ist die ausgewählte Katalogservicedomäne die Standarddomäne für die Zelle. (Boolean)
-userID	Gibt die Benutzer-ID für die Katalogservicedomäne an.
-password	Gibt das Kennwort für die Katalogservicedomäne an.
-properties	Gibt angepasste Eigenschaften für die Katalogservicedomäne.

Tabelle 22. Argumente für den Schritt "modifyEndpoints"

Argument	Beschreibung
-name	Gibt den Namen des Katalogserviceendpunkts an.
-hostName	Gibt den Namen des Hosts an, auf dem sich der Endpunkt befindet.
-endPoints	Gibt die Portnummern für den Endpunkt der Katalogservicedomäne an.

Tabelle 23. Argumente für den Schritt "addEndpoints"

Argument	Beschreibung
-name	Gibt den Namen des Katalogserviceendpunkts an.
-hostName	Gibt den Namen des Hosts an, auf dem sich der Endpunkt befindet.
-endPoints	Gibt die Portnummern für den Endpunkt der Katalogservicedomäne an.
-properties	Gibt angepasste Eigenschaften für den Endpunkt der Katalogservicedomäne an.

Tabelle 24. Argumente für den Schritt "removeEndpoints"

Argument	Beschreibung
-name	Gibt den Namen des zu löschenden Katalogserviceendpunkts an.

Rückgabewert:

Beispielsyntax für den Stapelmodus

- Mit Jacl:

```
$AdminTask modifyXSDomain {-name TestDomain -userID newuser -password *****
-default false -modifyEndpoints {{cs1 xhost1.ibm.com "" 5502,2809,1099}}
-addEndpoints {{cs3 xhost3.ibm.com "" 5501,2809,1099}}
-removeEndpoints {{cs2}}}
```

- Mit Jython (Zeichenfolge):

```
AdminTask.modifyXSDomain('[-name TestDomain -userID newuser -password *****
-default false -modifyEndpoints [[cs1 xhost1.ibm.com "" 5502,2809,1099]]
-addEndpoints [[cs3 xhost3.ibm.com "" 5501,2809,1099]]
-removeEndpoints [[cs2]]]')
```

Beispielsyntax für den Dialogmodus:

- Mit Jacl:

```
$AdminTask modifyXSDomain {-interactive}
```

- Mit Jython (Zeichenfolge):

```
AdminTask.modifyXSDomain ('[-interactive]')
```

testXSDomainConnection

Der Befehl "testXSDomainConnection" testet die Verbindung zu einer Katalogservicedomäne.

Erforderliche Parameter:

-name
Gibt den Namen der Katalogservicedomäne an, zu der die Verbindung getestet werden soll.

Optionale Parameter

-timeout
Gibt an, wie lange maximal (in Sekunden) auf die Verbindung gewartet wird.

Rückgabewert: Wenn eine Verbindung hergestellt werden kann, gibt der Befehl `true` zurück. Andernfalls werden Informationen zum Verbindungsfehler zurückgegeben.

Beispielsyntax für den Stapelmodus

- Mit Jacl:

```
$Admintask testXSDomainConnection
```

- Mit Jython (Zeichenfolge):

```
AdminTask.testXSDomainConnection
```

Beispielsyntax für den Dialogmodus:

- Mit Jacl:

```
$AdminTask testXSDomainConnection {-interactive}
```

- Mit Jython (Zeichenfolge):

```
AdminTask.testXSDomainConnection ('[-interactive]')
```

testXSServerConnection

Der Befehl "testXSServerConnection" testet die Verbindung zu einem Katalogserver. Dieser Befehl funktioniert für eigenständige Server und für Server, die zu einer Katalogservicedomäne gehören.

Erforderliche Parameter:

host
Gibt den Host an, auf dem sich der Katalogserver befindet.

listenerPort
Gibt den Listener-Port für den Katalogserver an.

Optionale Parameter

timeout
Gibt an, wie lange maximal (in Sekunden) auf eine Verbindung zum Katalogserver gewartet wird.

Rückgabewert:

Beispielsyntax für den Stapelmodus

- Mit Jacl:
  ```
  $Admintask testXSServerConnection {-host xhost1.ibm.com -listenerPort 2809}
  ```
- Mit Jython (Zeichenfolge):
  ```
  AdminTask.testXSServerConnection('[-host xshost3.ibm.com -listenerPort 2809]')
  ```

Beispielsyntax für den Dialogmodus:

- Mit Jacl:
  ```
  $AdminTask testXSServerConnection {-interactive}
  ```
- Mit Jython (Zeichenfolge):
  ```
  AdminTask.testXSServerConnection ('[-interactive]')
  ```

eXtreme-Scale-Container automatisch in Anwendungen von WebSphere Application Server starten

Containerprozesse in einer Umgebung mit WebSphere Application Server werden automatisch gestartet, wenn ein Modul gestartet wird, in dem die XML-Dateien von eXtreme Scale enthalten sind.

Vorbereitende Schritte

WebSphere Application Server und WebSphere eXtreme Scale müssen installiert und in der Lage sein, auf die Administrationskonsole von WebSphere Application Server zuzugreifen.

Informationen zu diesem Vorgang

Java-EE-Anwendungen haben komplexe Regeln für Klassenladeprogramme, die das Laden von Klassen erheblich komplexer machen, wenn eine gemeinsame Konfiguration von WebSphere eXtreme Scale in einem Java-EE-Server verwendet wird. Eine Java-EE-Anwendung ist gewöhnlich eine einzelne EAR-Datei, in der eine oder mehrere EJB- oder WAR-Module implementiert sind.

WebSphere eXtreme Scale überwacht den Start jedes Moduls und sucht nach XML-Dateien von eXtreme Scale. Wenn WebSphere eXtreme Scale feststellt, dass ein Modul mit den XML-Dateien gestartet wird, wird der Anwendungsserver als eXtreme-Scale-Container-JVM beim Katalogservice registriert. Durch die Registrierung der Container beim Katalogservice kann dieselbe Anwendung in mehreren Grids implementiert und trotzdem vom Katalogservice als einzelnes Grid behandelt werden. Der Katalogservice kümmert sich nicht um Zellen, Grids oder dynamische Grids. Die einzige Unterscheidung, die der Katalogservice trifft, ist die, ob eine

Komponente eine eXtreme-Scale-Container-JVM ist oder nicht. Ein einzelnes logisches Grid kann sich bei Bedarf über mehrere Zellen von WebSphere Extended Deployment erstrecken.

Vorgehensweise

1. Packen Sie Module in Ihre EAR-Datei, die die XML-Dateien für eXtreme Scale im Ordner META-INF enthalten. WebSphere eXtreme Scale erkennt das Vorhandensein der Dateien `objectGrid.xml` und `objectGridDeployment.xml` im Ordner META-INF von EJB- und WEB-Modulen, wenn diese gestartet werden. Wenn nur eine Datei `objectGrid.xml` gefunden wird, wird davon ausgegangen, dass die JVM ein Client ist. Sind beide Dateien vorhanden, wird davon ausgegangen, dass diese JVM als Container für das Grid agiert, das in der Datei `objectGridDeployment.xml` definiert ist.

   Sie müssen die richtigen Namen für diese XML-Dateien verwenden. Bei den Dateinamen muss die Groß-/Kleinschreibung beachtet werden. Wenn die Dateien nicht vorhanden sind, wird der Container nicht gestartet. Sie können in der Datei `systemout.log` nach Nachrichten suchen, die darauf hinweisen, dass Shards verteilt werden. Ein EJB-Modul oder WAR-Modul, das eXtreme Scale verwendet, muss XML-Dateien von eXtreme Scale in seinem Verzeichnis `META-INF` enthalten.

   Die XML-Dateien von eXtreme Scale enthalten Folgendes:

   - eine Datei `objectGrid.xml`, die allgemein auch als XML-Deskriptordatei von eXtreme Scale bezeichnet wird,
   - eine Datei `objectGridDeployment.xml`, die allgemein auch als XML-Implementierungsdeskriptordatei bezeichnet wird,
   - eine Datei `entity.xml`, falls Entitäten verwendet werden (optional). Der Name der Datei `entity.xml` muss mit dem Namen übereinstimmen, der in der Datei `objectGrid.xml` angegeben ist.

   Die Laufzeitumgebung von eXtreme Scale erkennt diese Dateien und stellt dann eine Verbindung zum Katalogservice her, um ihn darüber zu informieren, dass ein weiterer Container verfügbar ist, der Shards für diese Instanz von eXtreme Scale aufnehmen kann.

   Tipp: Wenn Sie eine Anwendung mit Entitäten unter Verwendung eines einzigen eXtreme-Scale-Servers verwenden möchten, setzen Sie die Eigenschaft "minSyncReplicas" in der XML-Implementierungsdeskriptordatei auf `0`. Andernfalls können Sie eine der folgenden Nachrichten in der Datei `SystemOut.log` sehen, weil keine Verteilung stattfinden kann, bis ein anderer Server der minSyncReplica-Richtlinie entspricht:

   ```
   CWPRJ1005E: Fehler beim Auflösen der Entitätsassoziation. Entity=entity_name,
   association=association_name.
   ```

   ```
   CWOBJ3013E: Das EntityMetadata-Repository ist nicht verfügbar. Beim Versuch, die Entität
   zu registrieren, wurde das zulässige Zeitlimit erreicht: Entitätsname
   ```

2. Implementieren und starten Sie Ihre Anwendung.

   Der Container wird automatisch gestartet, wenn das Modul gestartet wird. Der Katalogservice beginnt sobald wie möglich mit der Verteilung der primären und Replikat-Shards der Partition. Diese Verteilung findet sofort statt, sofern Sie kein Attribut "numInitialContainers" in der Datei `objectGridDeployment.xml` definieren. Wenn Sie das Attribut "numInitialContainers" definieren, findet die Verteilung statt, wenn die angegebene Anzahl an Containern gestartet wurde.

Nächste Schritte

Anwendungen, die sich in derselben Zelle wie die Container befinden, können eine Verbindung zu diesen Grid über die Methode "ObjectGridManager.connect(null, null) herstellen und anschließend die Methode "getObjectGrid(ccc, "object grid name")" aufrufen. Die Methoden "connect" und "getObjectGrid" können blockiert werden, bis die Container die Shards verteilt haben, aber diese Blockierung tritt nur auf, wenn das Grid gestartet wird.

ClassLoaders

Alle Plug-ins oder Objekte, die in eXtreme Scale gespeichert sind, werden in ein bestimmtes Klassenladeprogramm geladen. Zwei EJB-Module in derselben EAR-Datei können diese Objekte enthalten. Die Objekte sind identisch, werden aber von verschiedenen Klassenladeprogrammen geladen. Wenn Anwendung A ein Person-Objekt in einer eXtreme-Scale-Map, die eine lokale Map des Servers ist, empfängt Anwendung B eine Ausnahme des Typs "ClassCastException", wenn sie versucht, das Objekt zu lesen. Diese Ausnahme tritt ein, weil Anwendung B das Person-Objekt in einem anderen Klassenladeprogramm geladen hat.

Eine Methode zur Behebung dieses Problems ist die Verwendung eines Stammmoduls, das die erforderlichen Plug-ins und Objekte enthält, die in eXtreme Scale gespeichert werden. Jedes Modul, das eXtreme Scale verwendet, muss dieses Modul für seine Klassen referenzieren. Eine andere Lösung ist die Verwaltung dieser gemeinsam genutzten Objekte in einer Dienstprogramm-JAR-Datei, die in einem Klassenladeprogramm enthalten ist, das von Modulen und Anwendungen gemeinsam genutzt wird. Die Objekte können auch in WebSphere-Klassen oder in ein Verzeichnis `lib/ext` gestellt werden, aber dies macht die Implementierung komplexer und wird deshalb nicht empfohlen.

EJB-Module in einer EAR-Datei nutzen gewöhnlich dasselbe Klassenladeprogramm und sind von diesem Problem nicht betroffen. Jedes WAR-Modul hat ein eigenes Klassenladeprogramm und ist von diesem Problem betroffen.

Verbindung zu einem Grid als reiner Client herstellen

Wenn die Eigenschaft "catalog.services.cluster" in den angepassten Eigenschaften für die Zelle, den Knoten oder den Server definiert ist, kann jedes Modul in der EAR-Datei die Methode "ObjectGridManager.connect (ServerFactory.getServerProperties().getCatalogServiceBootstrap(), null, null)" aufrufen, um ein ClientClusterContext-Objekt abzurufen, und die Methode "ObjectGridManager.getObjectGrid(ccc, "grid name")", um eine Referenz auf eXtreme Scale abzurufen. Wenn Anwendungsobjekte in Maps gespeichert werden, müssen Sie sicherstellen, dass diese Objekte in einem gemeinsamen Klassenladeprogramm vorhanden sind.

Java-SE-Clients und Clients außerhalb der Zelle können über den Bootstrap-IIOP-Port des Katalogservice (standardmäßig der Deployment Manager in WebSphere) ein ClientClusterContext-Objekt und anschließend wie gewöhnlich eine benannte Instanz von eXtreme Scale abrufen.

EntityManager

Der EntityManager hilft, weil die Tupel in den Maps und nicht die Anwendungsobjekte werden, damit das Klassenladeprogramm weniger Probleme hat, um die es sich kümmern muss. Plug-ins können jedoch ein Problem sein. Beachten Sie auch, das immer eine eXtreme-Scale-XML-Deskriptordatei für Clientkorrekturwerte erfor-

derlich ist, wenn eine Verbindung zu einer Instanz von eXtreme Scale hergestellt wird, in der Entitäten definiert sind: ObjectGridManager.connect("host:port[,host-:port], null, objectGridOverride) oder ObjectGridManager.connect(null, objectGridOverride).

Programmgesteuerte Verwaltung mit Managed Beans (MBeans)

Sie können verschiedene Typen von JMX-Beans (Java Management Extensions) verwenden, um Implementierungen zu verwalten und zu überwachen. Jede MBean verweist auf eine bestimmte Entität, wie z. B. eine Map, ein Daten-Grid, einen Server oder einen Service.

JMX-MBean-Schnittstellen und WebSphere eXtreme Scale

Jede MBean hat get-Methoden, die Attributwerte darstellen. Diese get-Methoden können nicht direkt über Ihr Programm aufgerufen werden. Die JMX-Spezifikation behandelt Attribute anders als Operationen. Sie können Attribute über die JMX-Konsole eines anderen Anbieters anzeigen, und Sie können Operationen in Ihrem Programm oder über eine JMX-Konsole eines anderen Anbieters durchführen.

Paket "com.ibm.websphere.objectgrid.management"

Eine Übersicht und detaillierte Programmierspezifikatinen für alle verfügbaren MBeans finden Sie in der generierten API-Dokumentation zum Paket com.ibm-.websphere.objectgrid.management.

Mit dem Tool "wsadmin" auf MBeans zugreifen

Sie können das in WebSphere Application Server bereitgestellte Dienstprogramm "wsadmin" verwenden, um auf MBean-Informationen zuzugreifen.

Führen Sie das Tool "wsadmin" im Verzeichnis "bin" Ihrer Installation von WebSphere Application Server aus. Im folgenden Beispiel wird eine Sicht der aktuellen Shard-Verteilung in einer dynamischen eXtreme-Scale-Umgebung abgerufen. Sie können wsadmin in jeder Installation ausführen, in der eXtreme Scale ausgeführt wird. Das Dienstprogramm "wsadmin" muss nicht im Katalogservice ausgeführt werden.

```
$ wsadmin.sh -lang jython
wsadmin>placementService = AdminControl.queryNames
 ("com.ibm.websphere.objectgrid:*,type=PlacementService")
wsadmin>print AdminControl.invoke(placementService,
 "listObjectGridPlacement","library ms1")

<objectGrid name="library" mapSetName="ms1">
  <container name="container-0" zoneName="DefaultDomain"
  hostName="host1.company.org" serverName="server1">
     <shard type="Primary" partitionName="0"/>
     <shard type="SynchronousReplica" partitionName="1"/>
  </container>
  <container name="container-1" zoneName="DefaultDomain"
  hostName="host2.company.org" serverName="server2">
     <shard type="SynchronousReplica" partitionName="0"/>
     <shard type="Primary" partitionName="1"/>
  </container>
  <container name="UNASSIGNED" zoneName="_ibm_SYSTEM"
  hostName="UNASSIGNED" serverName="UNNAMED">
    <shard type="SynchronousReplica" partitionName="0"/>
    <shard type="AsynchronousReplica" partitionName="0"/>
  </container>
</objectGrid>
```

Kapitel 8. Implementierungsumgebung sichern

WebSphere® eXtreme Scale kann den Datenzugriff sichern, unter anderem durch Integration mit externen Sicherheitsprovidern. Zu den Aspekten der Sicherheit gehören Authentifizierung, Berechtigung, Transportsicherheit, Grid-Sicherheit, lokale Sicherheit und JMX-(Mbean-)Sicherheit.

Lokale Sicherheit aktivieren

WebSphere eXtreme Scale stellt mehrere Sicherheitsendpunkte für die Integration angepasster Mechanismen bereit. Im lokalen Programmiermodell ist die Hauptsicherheitsfunktion Berechtigung. Authentifizierung wird nicht unterstützt. Sie müssen die Authentifizierung unabhängig von der bereits vorhandenen Authentifizierung in WebSphere Application Server durchführen. Es werden jedoch Plug-ins für das Anfordern und Validieren von Subject-Objekten bereitgestellt.

In den folgenden Abschnitten werden zwei Methoden beschrieben, mit denen Sie die lokale Sicherheit aktivieren können:

Mit einer ObjectGrid-XML-Datei können Sie eine ObjectGrid-Instanz definieren und die Sicherheit für diese Instanz aktivieren. Die Datei `secure-objectgrid-definition.xml`, die in der Musterunternehmensanwendung "ObjectGridSample" verwendet wird, wird im folgenden Beispiel gezeigt. Setzen Sie das Attribut "securityEnabled" auf `true`, um die Sicherheit zu aktivieren.

```
<objectGrids>
    <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
        authorizationMechanism="AUTHORIZATION_MECHANISM_JAAS">
    ...
</objectGrids>
```

Sie können die Sicherheit auch über das Programm aktivieren. Zum Erstellen eines ObjectGrids mit der Methode "ObjectGrid.setSecurityEnabled" rufen Sie die folgende Methode in der Schnittstelle "ObjectGrid" auf:

```
/**
 * ObjectGrid-Sicherheit aktivieren
 */
void setSecurityEnabled();
```

Weitere Informationen finden Sie in der API-Dokumentation.

Grid-Authentifizierung

Sie können das sichere Token-Manager-Plug-in verwenden, um die Authentifizierung zwischen Servern zu aktivieren. Hierfür müssen Sie die Schnittstelle "SecureTokenManager" implementieren.

Die Methode "generateToken(Object)" verwendet ein Objekt und generiert anschließend ein Token, das von anderen nicht interpretiert werden kann. Die Methode "verifyTokens(byte[])" führt den umgekehrten Prozess aus. Sie konvertiert das Token zurück in das ursprüngliche Objekt.

Eine einfache SecureTokenManager-Implementierung verwendet einen einfachen Verschlüsselungsalgorithmus, wie z. B. einen XOR-Algorithmus (exklusives Oder), um das Objekt in serialisierter Form zu verschlüsseln, und anschließend den ent-

sprechenden Entschlüsselungsalgorithmus, um das Token zu entschlüsseln. Diese Implementierung ist nicht sicher und anfällig für Attacken.

Standardimplementierung von WebSphere eXtreme Scale

WebSphere eXtreme Scale stellt eine sofort verfügbare Implementierung für diese Schnittstelle bereit. Diese Standardimplementierung verwendet ein Schlüsselpaar, um die Signatur zu signieren und zu prüfen, und einen geheimen Schlüssel, um den Inhalt zu verschlüsseln. Jeder Server hat einen JCKES-Keystore, in dem das Schlüsselpaar (privater und öffentlicher Schlüssel) und der geheime Schlüssel gespeichert werden. Der Keystore muss ein JCKES-Keystore sein, damit geheime Schlüssel gespeichert werden können. Diese Schlüssel werden verwendet, um die Shared-Secret-Zeichenfolge auf Senderseite zu verschlüsseln und zu signieren bzw. zu prüfen. Außerdem wird dem Token eine Verfallszeit zugeordnet. Auf Empfängerseite werden die Daten geprüft, entschlüsselt und mit der Shared-Secret-Zeichenfolge des Empfängers verglichen. Es sind keine SSL-Kommunikationsprotokolle (Secure Sockets Layer) zwischen einem Serverpaar für die Authentifizierung erforderlich, weil die privaten und öffentlichen Schlüssel demselben Zweck dienen. Wenn die Serverkommunikation jedoch nicht verschlüsselt ist, können die Daten einfach durch Ansicht der Kommunikation gestohlen werden. Da das Token relativ bald verfällt, ist das Sicherheitsrisiko durch Attacken durch Nachrichtenaufzeichnung und -wiederholung minimal. Das Risiko ist erheblich geringer, wenn alle Server hinter einer Firewall implementiert werden.

Dieser Ansatz hat den Nachteil, dass die Administratoren von WebSphere eXtreme Scale Schlüssel generieren und an alle Server übermitteln müssen, was während des Transports der Schlüssel zu Sicherheitsverletzungen führen kann.

Grid-Sicherheit

Die Grid-Sicherheit von WebSphere eXtreme Scale gewährleistet, dass ein dem Grid beitretender Server die richtigen Berechtigungsnachweise besitzt, so dass kein zerstörerischer Server in das Grid aufgenommen wird. Die Grid-Sicherheit verwendet einen Mechanismus mit Shared-Secret-Zeichenfolgen.

Alle Server von WebSphere eXtreme Scale, einschließlich der Katalogserver, einigen sich auf eine Shared-Secret-Zeichenfolge. Wenn ein Server dem Grid beitritt, wird er aufgefordert, diese Shared-Secret-Zeichenfolge vorzulegen. Wenn die Shared-Secret-Zeichenfolge des beitretenden Servers mit der Zeichenfolge im führenden Server oder Katalogserver übereinstimmt, wird der beitretende Server akzeptiert, wenn nicht, wird die Join-Anforderung zurückgewiesen.

Das Senden einer Shared-Secret-Zeichenfolge als Klartext ist nicht sicher. Die Sicherheitsinfrastruktur von WebSphere eXtreme Scale stellt ein Manager-Plug-in für sichere Token bereit, damit der Server dieses Shared Secret vor dem Senden sichern kann. Sie müssen festlegen, wie die Sicherungsoperation implementiert wird. WebSphere eXtreme Scale stellt eine vordefinierte Implementierung bereit, in der die Sicherungsoperation so implementiert ist, dass das Shared Secret verschlüsselt und signiert wird.

Die Shared-Secret-Zeichenfolge wird in der Datei `server.properties` festgelegt. Weitere Informationen zur Eigenschaft "authenticationSecret" finden Sie im Abschnitt „Servereigenschaftendatei” auf Seite 188.

SecureTokenManager-Plug-in

Ein Manager-Plug-in für sichere Token wird über die Schnittstelle "com.ibm.websphere.objectgrid.security.plugins.SecureTokenManager" bereitgestellt.

Weitere Informationen zum SecureTokenManager-Plug-in finden Sie in der Dokumentation zur API 'SecureTokenManager'.

Die Methode "generateToken(Object)" verwendet ein Objekt und generiert anschließend ein Token, das von anderen nicht interpretiert werden kann. Die Methode "verifyTokens(byte[])" führt den umgekehrten Prozess aus. Sie konvertiert das Token zurück in das ursprüngliche Objekt.

Eine einfache SecureTokenManager-Implementierung verwendet einen einfachen Verschlüsselungsalgorithmus, wie z. B. einen XOR-Algorithmus (exklusives Oder), um das Objekt in serialisierter Form zu verschlüsseln, und anschließend den entsprechenden Entschlüsselungsalgorithmus, um das Token zu entschlüsseln. Diese Implementierung ist nicht sicher.

WebSphere eXtreme Scale stellt eine sofort verfügbare Implementierung für diese Schnittstelle bereit.

Die Standardimplementierung verwendet ein Schlüsselpaar, um die Signatur zu signieren und zu prüfen, und einen geheimen Schlüssel, um den Inhalt zu verschlüsseln. Jeder Server hat einen JCKES-Keystore, in dem das Schlüsselpaar (privater und öffentlicher Schlüssel) und der geheime Schlüssel gespeichert werden. Der Keystore muss ein JCKES-Keystore sein, damit geheime Schlüssel gespeichert werden können.

Diese Schlüssel werden verwendet, um die Shared-Secret-Zeichenfolge auf Senderseite zu verschlüsseln und zu signieren bzw. zu prüfen. Außerdem wird dem Token eine Verfallszeit zugeordnet. Auf Empfängerseite werden die Daten geprüft, entschlüsselt und mit der Shared-Secret-Zeichenfolge des Empfängers verglichen. Es sind keine SSL-Kommunikationsprotokolle (Secure Sockets Layer) zwischen einem Serverpaar für die Authentifizierung erforderlich, weil die privaten und öffentlichen Schlüssel demselben Zweck dienen. Wenn die Serverkommunikation jedoch nicht verschlüsselt ist, können die Daten einfach durch Ansicht der Kommunikation gestohlen werden. Da das Token relativ bald verfällt, ist das Sicherheitsrisiko durch Attacken durch Nachrichtenaufzeichnung und -wiederholung minimal. Das Risiko ist erheblich geringer, wenn alle Server hinter einer Firewall implementiert werden.

Dieser Ansatz hat den Nachteil, dass die eXtreme-Scale-Administratoren Schlüssel generieren und an alle Server übermitteln müssen, was während des Transports der Schlüssel zu Sicherheitsverletzungen führen kann.

Musterscripts zum Erstellen der Standardeigenschaften eines sicheren Tokenmanagers

Wie bereits im vorherigen Abschnitt erwähnt, können Sie einen Schlüsselspeicher erstellen, der ein Schlüsselpaar für die Signatur und Prüfung der Signatur sowie einen geheimen Schlüssel für die Verschlüsselung des Inhalts enthält.

Sie können beispielsweise den Befehl "keytool" von JDK 6 wie folgt verwenden, um die Schlüssel zu erstellen:

```
keytool -genkeypair -alias keypair1 -keystore key1.jck -storetype JCEKS -keyalg
rsa -dname "CN=sample.ibm.com, OU=WebSphere eXtreme Scale" -storepass key111 -keypass
keypair1 -validity 10000

keytool -genseckey -alias seckey1 -keystore key1.jck -storetype JCEKS -keyalg
DES  -storepass key111 -keypass seckey1 -validity 1000
```

Diese beiden Befehle erstellen ein Schlüsselpaar "keypair1" und einen geheimen Schlüssel "seckey1". Anschließend können Sie Folgendes in der Servereigenschaftendatei konfigurieren:

```
secureTokenKeyStore=key1.jck
secureTokenKeyStorePassword=key111
secureTokenKeyStoreType=JCEKS
secureTokenKeyPairAlias=keypair1
secureTokenKeyPairPassword=keypair1
secureTokenSecretKeyAlias=seckey1
secureTokenSecretKeyPassword=seckey1
secureTokenCipherAlgorithm=DES
secureTokenSignAlgorithm=RSA
```

Konfiguration

Weitere Informationen zu den Eigenschaften, mit denen Sie den Manager für sichere Token konfigurieren, finden Sie im Abschnitt Servereigenschaften.

Anwendungsclientauthentifizierung

Die Anwendungsclientauthentifizierung setzt sich aus der Aktivierung der Client/Server-Sicherheit, der Authentifizierung des Berechtigungsnachweises und der Konfiguration eines Authentifikators und eines Generators für Systemberechtigungsnachweise zusammen.

Client/Server-Sicherheit aktivieren

Für eine erfolgreiche Authentifizierung bei ObjectGrid müssen Sie die Sicherheit im Client und im Server aktivieren.

Clientsicherheit aktivieren

WebSphere eXtreme Scale stellt eine Musterclienteigenschaftendatei mit dem Namen `sampleClient.properties` im Verzeichnis *WAS_HOME*`/optionalLibraries/ObjectGrid/properties` für eine Installation von WebSphere Extended Deployment bzw. im Verzeichnis `/ObjectGrid/properties` für eine heterogene Serverinstallation bereit. Sie können diese Schablonendatei mit entsprechenden Werten anpassen. Setzen Sie die Eigenschaft "securityEnabled" in der Datei `objectgridClient.properties` auf `true`. Die Eigenschaft "securityEnabled" gibt an, ob die Sicherheit aktiviert ist. Wenn ein Client eine Verbindung zu einem Server herstellt, muss die Eigenschaft "securityEnabled" auf der Client- und auf der Serverseite denselben Wert haben: `true` oder `false`. Ist die Sicherheit beispielsweise im verbindungsherstellenden Server aktiviert, muss die Eigenschaft im Client auf `true` gesetzt werden, damit die Verbindung zum Server hergestellt werden kann.

Die Schnittstelle "com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration" stellt die Datei `security.ogclient.props` dar. Sie können die allgemein zugängliche Anwendungsprogrammierschnittstelle "com.ibm.websphere.objectgrid.security.config.ClientSecurityConfigurationFactory" verwenden, um eine Instanz dieser Schnittstelle mit Standardwerten zu erstellen. Sie können aber auch eine Instanz erstellen, indem Sie die Datei mit den Sicher-

heitseigenschaften des ObjectGrid-Clients übergeben. Die Datei `security.ogclient.props` enthält weitere Eigenschaften. Weitere Einzelheiten finden Sie in der Dokumentation zur API "ClientSecurityConfiguration" und in der Dokumentation zur API "ClientSecurityConfigurationFactory".

Serversicherheit aktivieren

Zum Aktivieren der Sicherheit auf der Serverseite können Sie die Eigenschaft **securityEnabled** in der Datei `security.xml` auf `true` setzen. Verwenden Sie eine XML-Sicherheitsdeskriptordatei, um die Daten-Grid-Sicherheitskonfiguration so zu definieren, das die Grid-weite Sicherheitskonfiguration von der Konfiguration, die sich nicht auf die Sicherheit bezieht, isoliert wird.

Authentifizierung des Berechtigungsnachweises

Nachdem der eXtreme-Scale-Client das Credential-Objekt mit dem CredentialGenerator-Objekt abgerufen hat, wird das Credential-Objekt zusammen mit der Clientanforderung an den eXtreme-Scale-Server gesendet. Der Server authentifiziert das Credential-Objekt, bevor er die Anforderung verarbeitet. Bei erfolgreicher Authentifizierung des Credential-Objekts wird ein Subject-Objekt zurückgegeben, dass dieses Credential-Objekt repräsentiert. Dieses Subject-Objekt wird anschließend für die Berechtigung der Anforderung verwendet.

Setzen Sie die Eigenschaft **credentialAuthentication** in den Client- und Servereigenschaftendateien so, dass die Authentifizierung des Berechtigungsnachweises aktiviert wird. Weitere Informationen finden Sie in den Abschnitten „Clienteigenschaftendatei” auf Seite 207 und „Servereigenschaftendatei” auf Seite 188.

Die folgende Tabelle enthält eine Übersicht über die für verschiedene Einstellungen zu verwendenden Authentifizierungsverfahren.

Tabelle 25. Authentifizierung des Berechtigungsnachweises bei Client- und Servereinstellungen

Authentifizierung des Clientberechtigungsnachweises	Authentifizierung des Serverberechtigungsnachweises	Ergebnis
Nein	Nie	Inaktiviert
Nein	Unterstützt	Inaktiviert
Nein	Erforderlich	Fehlersituation
Unterstützt	Nie	Inaktiviert
Unterstützt	Unterstützt	Aktiviert
Unterstützt	Erforderlich	Aktiviert
Erforderlich	Nie	Fehlersituation
Erforderlich	Unterstützt	Aktiviert
Erforderlich	Erforderlich	Aktiviert

Authentifikator konfigurieren

Der eXtreme-Scale-Server verwendet das Authenticator-Plug-in, um das Credential-Objekt zu authentifizieren. Eine Implementierung der Schnittstelle "Authenticator" ruft das Credential-Objekt ab und authentifiziert es dann bei einer Benutzer-Registry, z. B. einem LDAP-Server (Lightweight Directory Access Protocol) usw. eXtreme

Scale stellt keine Registry-Konfiguration bereit. Die Herstellung einer Verbindung zu und die Authentifizierung bei einer Benutzer-Registry muss in diesem Plug-in implementiert werden.

Eine Authenticator-Implementierung extrahiert beispielsweise die Benutzer-ID und das Kennwort aus dem Berechtigungsnachweis, verwendet diese Informationen für die Herstellung einer Verbindung zu einem und Validierung bei einem LDAP-Server und erstellt als Ergebnis der Authentifizierung ein Subject-Objekt. Die Implementierung kann JAAS-Anmeldemodule (Java Authentication and Authorization Service) verwenden. Als Ergebnis der Authentifizierung wird ein Subject-Objekt zurückgegeben.

Sie können den Authentifikator, wie im folgenden Beispiel gezeigt, in der XML-Sicherheitsdeskriptordatei konfigurieren:

```
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config/security ../objectGridSecurity.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config/security">

 <security securityEnabled="true"
        loginSessionExpirationTime="300">

  <authenticator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.KeyStoreLoginAuthenticator">
        </authenticator>

 </security>

</securityConfig>
```

Verwenden Sie die Option **-clusterSecurityFile** beim Starten eines sicheren Servers, um die XML-Sicherheitsdatei festzulegen. Weitere Informationen finden Sie im Lernprogramm zur Java-SE-sicherheit in der *Produktübersicht*.

Generator für Systemberechtigungsnachweise konfigurieren

Der Generator für Systemberechtigungsnachweise wird für die Darstellung einer Factory für die Systemberechtigungsnachweise verwendet. Ein Systemberechtigungsnachweis gleicht einem Administratorberechtigungsnachweis. Sie können das Element "SystemCredentialGenerator" in der XML für Katalogsicherheit konfigurieren, wie es im folgenden Beispiel gezeigt wird:

```
<systemCredentialGenerator className ="com.ibm.websphere.objectgrid.security.plugins.
 builtins.UserPasswordCredentialGenerator">
          <property name="properties" type="java.lang.String" value="manager manager1"
      description="username password" />
 </systemCredentialGenerator>
```

Zu Demonstrationszwecken werden Benutzername und Kennwort in Klartext gespeichert. Speichern Sie den Benutzernamen und das Kennwort in einer Produktionsumgebung nicht in Klartext.

WebSphere eXtreme Scale stellt einen Standardgenerator für Systemberechtigungsnachweise bereit, der die Serverberechtigungsnachweise verwendet. Wenn Sie den Generator für Systemberechtigungsnachweise nicht explizit angeben, wird dieser Standardgenerator für Systemberechtigungsnachweise verwendet.

Anwendungsclientberechtigung

Die Anwendungsclientberechtigung setzt sich aus ObjectGrid-Berechtigungsklassen, Berechtigungsmechanismen, einem Berechtigungsprüfintervall und dem Feature "Zugriff nur durch Ersteller" zusammen.

Bei eXtreme Scale basiert die Berechtigung auf dem Subject-Objekt und auf Berechtigungen. Das Produkt unterstützt zwei Arten von Berechtigungsmechanismen: Java Authentication and Authorization Service (JAAS) und angepasste Berechtigungen.

ObjectGrid-Berechtigungsklassen

Die Berechtigung basiert auf Berechtigungen. Es gibt die folgenden vier verschiedenen Typen von Berechtigungsklassen:

- Die Klasse "MapPermission" stellt Berechtigungen für den Zugriff auf die Daten in ObjectGrid-Maps dar.
- Die Klasse "ObjectGridPermission" stellt Berechtigungen für den Zugriff auf ObjectGrid dar.
- Die Klasse "ServerMapPermission" stellt Berechtigungen für den Zugriff auf ObjectGrid-Maps auf der Serverseite über einen Client dar.
- Die Klasse "AgentPermission" stellt Berechtigungen zum Starten eines Agenten auf der Serverseite dar.

Weitere Informationen zu APIs und zugehörigen Berechtigungen finden Sie im Abschnitt zur Programmierung der Clientberechtigung im *Programmierhandbuch*.

Berechtigungsprüfintervall

eXtreme Scale unterstützt das Caching der Berechtigungsprüfergebnisse für Leistungszwecke. Ist dieser Mechanismus nicht vorhanden und wird eine in der Liste der Methoden und der erforderlichen Berechtigungen aufgeführte Methode aufgerufen, ruft die Laufzeitumgebung den konfigurierten Berechtigungsmechanismus zur Berechtigung des Zugriffs auf. Wenn das Berechtigungsprüfintervall definiert ist, wird der Berechtigungsmechanismus auf der Basis des festgelegten Berechtigungsprüfintervalls in regelmäßigen Abständen aufgerufen.

Die Berechtigungsinformationen basieren auf dem Subject-Objekt. Wenn ein Client versucht, auf die Methoden zuzugreifen, durchsucht die Laufzeitumgebung von eXtreme Scale den Cache nach dem Subject-Objekt. Wird das Objekt nicht im Cache gefunden, prüft die Laufzeitumgebung die für dieses Subject-Objekt erteilten Berechtigungen und speichert dann die Berechtigungen in einem Cache.

Das Berechtigungsprüfintervall muss vor der Initialisierung von ObjectGrid definiert werden. Sie können das Berechtigungsprüfintervall auf zwei Arten konfigurieren:

Sie können die ObjectGrid-XML-Datei verwenden, um ein ObjectGrid zu definieren und das Berechtigungsprüfintervall festzulegen. Im folgenden Beispiel wird das Berechtigungsprüfintervall auf 45 Sekunden gesetzt:

```
<objectGrids>
 <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
 authorizationMechanism="AUTHORIZATION_MECHANISM_JAAS"
 permissionCheckPeriod="45">
  <bean id="bean id="TransactionCallback"
className="com.ibm.websphere.samples.objectgrid.HeapTransactionCallback" />
...
</objectGrids>
```

Wenn Sie ein ObjectGrid über APIs erstellen möchten, rufen Sie die folgende Methode auf, um das Berechtigungsprüfintervall festzulegen. Diese Methode kann nur vor der Initialisierung der ObjectGrid-Instanz aufgerufen werden. Sie gilt nur dann für das lokale eXtreme-Scale-Programmiermodell, wenn Sie die ObjectGrid-Instanz direkt instanziieren.

```
/**
 * Diese Methode akzeptiert einen einzigen Parameter, der angibt, wie oft
 * die Berechtigung geprüft werden soll, die verwendet wird, um einem
 * Clientzugriff zuzulassen. Wenn der Parameter den Wert 0 hat, wird
 * der Berechtigungsmechanismus (JAAS-Berechtigung oder angepasste
 * Berechtigung) bei jedem Aufruf von get/put/update/remove/evict
 * aufgefordert zu prüfen, ob das aktuelle Subject-Objekt die erforderlichen
 * Berechtigungen besitzt. Dies kann je nach Berechtigungsimplementierung
 * vom Leistungsstandpunkt aus gesehen untragbar kostenintensiv sein, aber
 * sollten Sie den Berechtigungsmechanismus jemals aufrufen müssen, dann
 * setzen Sie den Parameter auf 0.
* Wenn Sie den Parameter auf einen Wert > 0 setzen, gibt er an, wie lange
 * (in Sekunden) ein Berechtigungssatz zwischengespeichert werden soll, bevor
 * sie zur Aktualisierung an den Berechtigungsmechanismus zurückgegeben werden.
 * Dieser Wert liefert eine sehr viel bessere Leistung, aber wenn die
 * Back-End-Berechtigungen in dieser Zeit geändert werden, kann das
 * ObjectGrid den Zugriff zulassen oder verweigern, obwohl der
 * Back-End-Sicherheitsprovider geändert wurde.
*
 * @param Zeitraum - Das Berechtigungsprüfintervall in Sekunden
 */
void setPermissionCheckPeriod(int period);
```

Berechtigung "Zugriff nur durch Ersteller"

Die Berechtigung "Zugriff nur durch Ersteller" gewährleistet, dass ausschließlich der Benutzer (dargestellt durch die zugeordneten Principal-Objekte), der den Eintrag in die ObjectGrid-Map einfügt, auf diesen Eintrag zugreifen (lesen, aktualisieren, ungültig machen und entfernen) kann.

Das vorhandene Berechtigungsmodell für ObjectGrid-Maps basiert auf dem Zugriffstyp, aber nicht auf Dateneinträgen. Anders ausgedrückt, ein Benutzer kann mit einem bestimmten Zugriffstyp (z. B. lesen, schreiben, einfügen, löschen oder ungültig machen) entweder auf alle Daten in der Map oder auf keine Daten in der Map zugreifen. eXtreme Scale berechtigt Benutzer jedoch nicht für den Zugriff auf einzelne Dateneinträge. Dieses Feature bietet eine neue Methode für die Berechtigung von Benutzern für den Zugriff auf Dateneinträge.

In einem Szenario, in dem verschiedene Benutzer auf verschiedene Datengruppen zugreifen, kann dieses Modell hilfreich sein. Wenn der Benutzer Daten aus dem persistenten Speicher in die ObjectGrid-Maps lädt, kann der Zugriff vom persistenten Speicher berechtigt werden. In diesem Fall muss keine weitere Berechtigung auf der Ebene der ObjectGrid-Maps erfolgen. Sie müssen lediglich sicherstellen, dass die Person, die die Daten in die Map lädt, auf die Map zugreifen kann, indem Sie das Feature "Zugriff nur durch Ersteller" aktivieren.

Werte für das Attribut "accessByCreatorOnlyMode":

disabled
Das Feature "Zugriff nur durch Ersteller" ist inaktiviert.

complement
Das Feature "Zugriff nur durch Ersteller" ist aktiviert, um die Map-Berechtigung zu ergänzen. In anderen Worten, die Map-Berechtigung und das Feature "Zugriff nur durch Ersteller" sind wirksam. Deshalb können Sie die Datenoperationen weiter einschränken. Der Ersteller kann die Daten beispielsweise nicht ungültig machen.

supersede
Das Feature "Zugriff nur durch Ersteller" ist aktiviert, um die Map-Berechtigung außer Kraft zu setzen. In anderen Worten, das Feature "Zugriff nur durch Ersteller" setzt die Map-Berechtigung außer Kraft, d. h., es wird keine Map-Berechtigung durchgeführt.

Sie können den Modus für das Feature "Zugriff nur durch Ersteller" auf zwei Arten konfigurieren:

Mit einer XML-Datei:

Sie können die ObjectGrid-XML-Datei verwenden, um ein ObjectGrid zu definieren und den Modus für das Feature "Zugriff nur durch Ersteller" auf `disabled` (Inaktiviert), `complement` (Ergänzung) oder `supersede` (Überlagern) zu setzen,

```
<objectGrids>
    <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
         accessByCreatorOnlyMode="supersede"
        <bean id="TransactionCallback"
              classname="com.ibm.websphere.samples.objectgrid.HeapTransactionCallback" />
    ...
</objectGrids>
```

Programmgesteuert:

Wenn Sie ein ObjectGrid über das Programm erstellen möchten, können Sie die folgende Methode aufrufen, um den Modus für das Feature "Zugriff nur durch Ersteller" festzulegen. Der Aufruf dieser Methode gilt nur dann für das lokale eXtreme-Scale-Programmiermodell, wenn Sie die ObjectGrid-Instanz direkt instanziieren:

```
/**
 * Legen Sie den Modus für das Feature "Zugriff nur durch Ersteller"
 * (ACCESS_BY_CREATOR_ONLY) fest.
 * Wenn Sie das Feature "Zugriff nur durch Ersteller" aktivieren, kann nur
 * der Benutzer (dargestellt durch die zugeordneten Principals), der den
 * Datensatz in die Map einfügt, auf den Datensatz zugreifen (lesen,
 * aktualisieren, ungültig machen und entfernen).
* Das Feature "Zugriff nur durch Ersteller"  kann inaktiviert werden
 * oder das ObjectGrid-Berechtigungsmodell ergänzen oder sogar außer
 * Kraft setzen. Standardmäßig ist das Feature inaktiviert:
 * {@link SecurityConstants#ACCESS_BY_CREATOR_ONLY_DISABLED}.
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_DISABLED
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_COMPLEMENT
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_SUPERSEDE
 *
 * @param accessByCreatorOnlyMode the access by creator mode.
 *
 * @since WAS XD 6.1 FIX3
*/
void setAccessByCreatorOnlyMode(int accessByCreatorOnlyMode);
```

Zur weiteren Veranschaulichung stellen Sie sich ein Szenario vor, in dem eine ObjectGrid-Map "account" ein Banken-Grid ist und Manager1 und Employee1 zwei Benutzer sind. Die eXtreme-Scale-Berechtigungsrichtlinie erteilt "Manager1" alle Zugriffsberechtigungen, "Employee1" aber nur Lesezugriff. Die JAAS-Richtlinie für die ObjectGrid-Map-Berechtigung ist im folgenden Beispiel gezeigt:

```
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
    Principal com.acme.PrincipalImpl "Manager1" {
    permission com.ibm.websphere.objectgrid.security.MapPermission
        "banking.account", "all"
};
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
    Principal com.acme.PrincipalImpl "Employee1" {
    permission com.ibm.websphere.objectgrid.security.MapPermission
        "banking.account", "read, insert"
};
```

Machen Sie sich Gedanken darüber, wie sich das Feature "Zugriff nur durch Ersteller" auf die Berechtigung auswirkt:

- **Inaktiviert:** Wenn das Feature "Zugriff nur durch Ersteller" inaktiviert ist, sind keine Auswirkungen auf die Map-Berechtigung zu verzeichnen. Der Benutzer "Manager1" kann auf alle Daten in der Map "account" zugreifen. Der Benutzer "Employee1" kann alle Daten in der Map lesen, aber keine Daten in der Map aktualisieren, ungültig machen oder entfernen.
- **Ergänzung:** Wenn Sie das Feature "Zugriff nur durch Ersteller" mit der Option "complement" (Ergänzung) aktivieren, sind die Map-Berechtigung und die Berechtigung über das Feature "Zugriff nur durch Ersteller" wirksam. Der Benutzer

"Manager1" kann auf die Daten in der Map "account" zugreifen, aber nur dann, wenn ausschließlich er sie in die Map geladen hat. Der Benutzer "Employee1" kann die Daten in der Map "account" lesen, aber nur dann, wenn auschließlich er sie in die Map geladen hat. (Dieser Benutzer kann jedoch keine Daten in der Map aktualisieren, ungültig machen oder entfernen.)

- **Überlagern:** Wenn Sie das Feature "Zugriff nur durch Ersteller" mit der Option "supersede" (Überlagern) aktivieren, wird die Map-Berechtigung nicht umgesetzt. Die Berechtigung über das Feature "Zugriff nur durch Ersteller" ist die einzige Berechtigungsrichtlinie. Der Benutzer "Manager1" hat dieselben Privilegien wie im Modus "Ergänzung". Er kann nur auf die Daten in der Map "account" zugreifen, wenn er selbst die Daten in die Map geladen hat. Der Benutzer "Employee1" hat jedoch vollständigen Zugriff auf die Daten in der Map "account", wenn er selbst die Daten in die Map geladen hat. Anders ausgedrückt, die in der JAAS-Richtlinie definierte Berechtigungsrichtlinie wird nicht umgesetzt.

Transport Layer Security und Secure Sockets Layer

WebSphere eXtreme Scale unterstützt TCP/IP und Transport Layer Security/Secure Sockets Layer (TLS/SSL) für die sichere Kommunikation zwischen Clients und Servern.

TLS/SSL unterstützt die sichere Kommunikation zwischen Client und Server. Der verwendete Kommunikationsmechanismus richtet sich nach dem Wert des Parameters "transportType", der in den Konfigurationsdateien von Client und Server angegeben ist.

Sie können die Eigenschaft "transportType" in den folgenden Client- und Serverkonfigurationsdateien definieren.

- Informationen zum Definieren der Eigenschaft in der Clientsicherheitskonfiguration finden Sie im Abschnitt „Clienteigenschaftendatei” auf Seite 207.
- Informationen zum Definieren der Eigenschaft in der Sicherheitskonfiguration des Containerservers finden Sie im Abschnitt „Servereigenschaftendatei” auf Seite 188.
- Informationen zum Definieren der Eigenschaft in der Sicherheitskonfiguration des Katalogservers finden Sie im Abschnitt „Servereigenschaftendatei” auf Seite 188.

Tabelle 26. Für bestimmte Clienttransport- und Servertransporteinstellungen zu verwendendes Protokoll

Eigenschaft "transportType" des Clients	Eigenschaft "transportType" des Servers	Zu verwendendes Protokoll
TCP/IP	TCP/IP	TCP/IP
TCP/IP	SSL-supported	TCP/IP
TCP/IP	SSL-required	Fehler
SSL-supported	TCP/IP	TCP/IP
SSL-supported	SSL-supported	SSL (wenn SSL scheitert, dann TCP/IP)
SSL-supported	SSL-required	SSL
SSL-required	TCP/IP	Fehler
SSL-required	SSL-supported	SSL
SSL-required	SSL-required	SSL

Wenn SSL verwendet wird, müssen die SSL-Konfigurationsparameter auf Client- und auf Serverseite angegeben werden. In einer Java-SE-Umgebung wird SSL in den Client- und Servereigenschaftendateien konfiguriert. Wenn sich der Client oder

Server in einem WebSphere Application Server befindet, können Sie die Unterstützung der Transportsicherheit in WebSphere Application Server verwenden, um SSL-Parameter zu konfigurieren.

Datei `orb.properties` für die Unterstützung der Transportsicherheit konfigurieren

Sie können TLS/SSL verwenden, wenn die Eigenschaft "transportType" den Wert "SSL-Supported" hat.

Zur Unterstützung sicherer Transporte in einer Java-SE-Umgebung müssen Sie die Datei „ORB-Eigenschaftendatei" auf Seite 197 ändern und die folgenden Eigenschaften einfügen:

```
# Eigenschaften des IBM JDK
org.omg.CORBA.ORBClass=com.ibm.CORBA.iiop.ORB
org.omg.CORBA.ORBSingletonClass=com.ibm.rmi.corba.ORBSingleton
javax.rmi.CORBA.StubClass=com.ibm.rmi.javax.rmi.CORBA.StubDelegateImpl
javax.rmi.CORBA.PortableRemoteObjectClass=com.ibm.rmi.javax.rmi.PortableRemoteObject
javax.rmi.CORBA.UtilClass=com.ibm.ws.orb.WSUtilDelegateImpl

# WS-Plug-ins
com.ibm.CORBA.ORBPluginClass.com.ibm.ws.orbimpl.transport.WSTransport
com.ibm.CORBA.ORBPluginClass.com.ibm.ws.orbimpl.WSORBPropertyManager
com.ibm.CORBA.ORBPluginClass.com.ibm.ISecurityUtilityImpl.SecurityPropertyManager

# WS-Interceptor
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.objectgrid.corba.ObjectGridInitializer
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ISecurityLocalObjectBaseL13Impl.SecurityComponentFactory

# Eigenschaften des WS-ORB und der Plug-ins
com.ibm.ws.orb.transport.ConnectionInterceptorName=com.ibm.ISecurityLocalObjectBaseL13Impl.SecurityConnectionInterceptor
com.ibm.ws.orb.transport.WSSSLClientSocketFactoryName=com.ibm.ws.security.orbssl.WSSSLClientSocketFactoryImpl

com.ibm.CORBA.TransportMode=Pluggable
com.ibm.CORBA.ServerName=ogserver
```

SSL-Parameter für eXtreme-Scale-Clients konfigurieren

Sie können SSL-Parameter für Clients wie folgt konfigurieren:

1. Erstellen Sie mit der Factory-Klasse "com.ibm.websphere.objectgrid.security.config. ClientSecurityConfigurationFactory" ein com.ibm.websphere.objectgrid.security.config.SSLConfiguration-Objekt. Weitere Einzelheiten finden Sie in der API-Dokumentation zu ClientSecurityConfigurationFactory.
2. Konfigurieren Sie die Parameter in der Datei `client.propeties`, und verwenden Sie anschließend die Methode "ClientSecurityConfigurationFactory.getClientSecurityConfiguration(String)", um die Objektinstanz mit Daten zu füllen.

Beispiele für Eigenschaften, die Sie in einem Client definieren können, finden Sie in der Beschreibung der Sicherheitseigenschaften des Clients im Abschnitt „Clienteigenschaftendatei" auf Seite 207.

SSL-Parameter für eXtreme-Scale-Server konfigurieren

Die SSL-Parameter für Server werden in einer Servereigenschaftendatei wie der Beispieldatei `server.properties` konfiguriert. Diese Eigenschaftendatei kann beim Starten eines eXtreme-Scale-Servers als Parameter übergeben werden. Weitere Informationen zu den SSL-Parametern, die Sie für eXtreme-Scale-Server definieren können finden Sie im Abschnitt „Servereigenschaftendatei" auf Seite 188.

Unterstützung der Transportsicherheit in WebSphere Application Server

Wenn ein eXtreme-Scale-Client, -Containerserver oder -Katalogserver in einem Prozess von WebSphere Application Server ausgeführt wird, wird die eXtreme-Scale-Transportsicherheit über die CSIV2-Transporteinstellungen verwaltet. Für den eXtreme-Scale-Client oder -Containerserver sollten Sie die SSL-Einstellungen nicht über die Eigenschaften des eXtreme-Scale-Clients oder -Servers konfigurieren. Alle SSL-Einstellungen sollten in der Konfiguration von WebSphere Application Server angegeben werden.

Der Katalogserver ist jedoch ein wenig anders. Der Katalogserver hat eigene proprietäre Transportpfade, die nicht über die CSIV2-Transporteinstellungen von Application Server verwaltet werden können. Deshalb müssen die SSL-Eigenschaften weiterhin in der Servereigenschaftendatei für den Katalogserver konfiguriert werden.

Transportsicherheit für Sun JDK aktivieren

WebSphere eXtreme Scale erfordert IBM Java Secure Sockets Extension (IBMJSSE) oder IBM Java Secure Sockets Extension 2 (IBMJSSE2). Die Provider IBMJSSE und IBMJSSE2 enthalten eine Referenzimplementierung, die die Protokolle SSL und TLS (Transport Layer Security) und ein API-Framework unterstützt.

Im reinen Sun JDK werden die Provider IBM JSSE und IBM JSSE2 nicht mitgeliefert. Deshalb kann die Transportsicherheit mit einem Sun JDK nicht aktiviert werden. Für die Aktivierung der Transportsicherheit ist ein mit WebSphere Application Server geliefertes Sun JDK erforderlich. Das mit WebSphere Application Server gelieferte Sun JDK enthält die Provider IBM JSSE und IBM JSSE2.

Lesen Sie die Informationen zum Konfigurieren eines Object Request Broker, um ein nicht von IBM stammendes JDK für WebSphere eXtreme Scale zu verwenden. Wenn `-Djava.endorsed.dirs` konfiguriert ist, zeigt diese Eigenschaft auf die Verzeichnisse `objectgridRoot/lib/endorsed` und `JRE/lib/endorsed`. Das Verzeichnis `objectgridRoot/lib/endorsed` ist erforderlich, damit der IBM ORB verwendet wird, und das Verzeichnis `JRE/lib/endorsed` ist erforderlich, um die Provider IBM JSSE und IBM JSSE2 zu laden.

In Schritt 4 des Lernprogramms zur Sicherheit in der *Produktübersicht* finden Sie Informationen zur Konfiguration der erforderlichen SSL-Eigenschaften, zum Erstellen von Keystores und Truststores und zum Starten sicherer Server in WebSphere eXtreme Scale.

JMX-Sicherheit (Java Management Extensions)

Sie können MBean-Aufrufe (Managed Beans) in einer verteilten Umgebung sichern.

Weitere Informationen zu den verfügbaren MBeans finden Sie im Abschnitt „Programmgesteuerte Verwaltung mit Managed Beans (MBeans)” auf Seite 360.

In der verteilten Implementierungstopologie befinden sich MBeans direkt in den Katalogservern und Containerservern. Im Allgemeinen folgt die JMX-Sicherheit in einer verteilten Topologie der JMX-Sicherheitsspezifikation, die in der Spezifikation "Java Management Extensions" festgelegt ist. Sie setzt sich aus den folgenden drei Komponenten zusammen:

1. Authentifizierung: Der ferne Client muss im Connector-Server authentifiziert werden.
2. Zugriffssteuerung: Die MBean-Zugriffssteuerung legt fest, wer auf die MBean-Informationen zugreifen und wer die MBean-Operationen durchführen kann.
3. Sicherer Transport: Der Transport zwischen dem JMX-Client und dem Server kann mit TLS/SSL gesichert werden.

Authentifizierung

JMX stellt den Connector-Servern Methoden für die Authentifizierung der fernen Clients zur Verfügung. Für den RMI-Connector wird die Authentifizierung durchgeführt, indem beim Erstellen des Connector-Servers ein Objekt übergeben wird, das die Schnittstelle "JMXAuthenticator" implementiert. Deshalb implementiert eXtreme Scale die Schnittstelle "JMXAuthenticator", um das ObjectGrid-Authenticator-Plug-in zu nutzen, um die fernen Clients zu authentifizieren. In dem Lernprogramm zur Sicherheit in der *Produktübersicht* wird detailliert beschrieben, wie eXtreme Scale einen Client authentifiziert.

Der JMX-Client folgt den JMX-APIs, um die Berechtigungsnachweise für die Verbindungsherstellung zum Connector-Server bereitzustellen. Das JMX-Framework übergibt die Berechtigungsnachweise an den Connector-Server und ruft dann die JMXAuthenticator-Implementierung für die Authentifizierung auf. Wie zuvor beschrieben, delegiert die JMXAuthenticator-Implementierung die Authentifizierung an die ObjectGrid-Authenticator-Implementierung.

Sehen Sie sich das folgende Beispiel an, das veranschaulicht, wie mit einem Berechtigungsnachweis eine Verbindung zu einem Connector-Server hergestellt wird:

```
javax.management.remote.JMXServiceURL jmxUrl = new JMXServiceURL(
        "service:jmx:rmi:///jndi/rmi://localhost:1099/objectgrid/MBeanServer");

    environment.put(JMXConnector.CREDENTIALS, new UserPasswordCredential("admin", "xxxxxx"));

    // JMXConnectorServer erstellen
    JMXConnector cntor = JMXConnectorFactory.newJMXConnector(jmxUrl, null);

    // Verbindung herstellen und eine Operation im fernen MBeanServer aufrufen
    cntor.connect(environment);
```

Im vorherigen Beispiel wird ein UserPasswordCredential-Objekt mit der Benutzer-ID "admin" und dem Kennwort "xxxxx" bereitgestellt. Dieses UserPasswordCredential-Objekt wird in der Umgebungs-Map gesetzt, die von der Methode "JMXConnector.connect(Map)" verwendet wird. Anschließend wird dieses UserPasswordCredential-Objekt über das JMX-Framework zunächst an den Server und schließlich zur Authentifizierung an das ObjectGrid-Authentifizierungs-Framework übergeben.

Das Clientprogrammiermodell folgt strikt der JMX-Spezifikation.

Zugriffssteuerung

Ein JMX-MBean-Server kann Zugriff auf sensible Informationen haben und deshalb in der Lage sein, sensible Operationen durchzuführen. JMX stellt die erforderliche Zugriffssteuerung bereit, die feststellt, welche Clients auf diese Informationen zugreifen und welche Clients diese Operationen durchführen dürfen. Die Zugriffssteuerung wird in das Java-Standardsicherheitsmodell integriert, indem Berechtigungen definiert werden, die den Zugriff auf den MBean-Server und seine Operationen steuern.

Bei der Zugriffssteuerung und -berechtigung für JMX-Operationen stützt sich eXtreme Scale auf die JAAS-Unterstützung, die die JMX-Implementierung bereitstellt. Zu jedem beliebigen Zeitpunkt während der Ausführung eines Programms besitzt ein Ausführungs-Thread einen aktuellen Satz an Berechtigungen. Wenn ein solcher Thread eine Operation der JMX-Spezifikation aufruft, handelt es sich um die so genannten "gehaltenen Berechtigungen". Bei der Durchführung einer JMX-Operation wird eine Sicherheitsprüfung durchgeführt, um festzustellen, ob die "gehaltenen Berechtigungen" die erforderliche Berechtigung abdecken.

Die MBean-Richtliniendefinition folgt dem Java-Richtlinienformat. Die folgende Richtlinie erteilt beispielsweise allen Unterzeichnern und allen Codebasen das Recht, die JMX-Adresse des Servers für die MBean "PlacementServiceMBean" abzurufen, allerdings mit Einschränkung auf die Domäne "com.ibm.websphere.objectgrid":

```
grant {
    permission javax.management.MBeanPermission
        "com.ibm.websphere.objectgrid.management.PlacementServiceMBean#retrieveServerJMXAddress
     [com.ibm.websphere.objectgrid:*,type=PlacementService]",
        "invoke";
}
```

Sie können das folgende Richtlinienbeispiel verwenden, um die Berechtigung auf der Basis der Identität des fernen Clients durchzuführen. Die Richtlinie erteilt dieselben MBean-Berechtigungen wie im vorherigen Beispiel, aber nur den Benutzern mit dem X500Principal-Namen "CN=Administrator,OU=software,O=IBM,L=Rochester,ST=MN,C=US".

```
grant principal javax.security.auth.x500.X500Principal "CN=Administrator,OU=software,O=IBM,
    L=Rochester,ST=MN,C=US" {permission javax.management.MBeanPermission
        "com.ibm.websphere.objectgrid.management.PlacementServiceMBean#retrieveServerJMXAddress
    [com.ibm.websphere.objectgrid:*,type=PlacementService]",
        "invoke";
}
```

Java-Richtlinien werden nur geprüft, wenn der Sicherheitsmanager aktiviert ist. Starten Sie Katalogserver und Containerserver mit dem JVM-Argument "-Djava.security.manager", um die Zugriffssteuerung für MBean-Operationen umzusetzen.

Sicherer Transport

Der Transport zwischen dem JMX-Client und dem Server kann mit TLS/SSL gesichert werden. Wenn das Attribut "transportType" des Katalogserver oder Containerservers auf "SSL_Required" oder "SSL_Supported" gesetzt ist, müssen Sie SSL verwenden, um die Verbindung zum JMX-Server herzustellen.

Zur Verwendung von SSL müssen Sie den Truststore, den Truststore-Typ und das Truststore-Kennwort im MBean-Client mit Systemeigenschaften konfigurieren, die mit "-D" beginnen:

1. -Djavax.net.ssl.trustStore=TRUST_STORE_LOCATION
2. -Djavax.net.ssl.trustStorePassword=TRUST_STORE_PASSWORD
3. -Djavax.net.ssl.trustStoreType=TRUST_STORE_TYPE

Wenn Sie "com.ibm.websphere.ssl.protocol.SSLSocketFactory" als SSL-Socket-Factory in der Datei "JAVA_HOME/jre/lib/security/java.security" verwenden, definieren Sie die folgenden Eigenschaften:

1. -Dcom.ibm.ssl.trustStore=TRUST_STORE_LOCATION
2. -Dcom.ibm.ssl.trustStorePassword=TRUST_STORE_PASSWORD
3. -Dcom.ibm.ssl.trustStoreType=TRUST_STORE_TYPE

Sicherheitsintegration mit externen Providern

Zum Schutz Ihrer Daten in WebSphere eXtreme Scale kann eXtreme Scale mit mehreren Sicherheitsprovidern integriert werden.

WebSphere eXtreme Scale kann mit einer externen Sicherheitsimplementierung integriert werden. Diese externe Implementierung muss Authentifizierungs- und Berechtigungsservices für eXtreme Scale bereitstellen. eXtreme Scale hat Plug-in-Punkte für die Integration einer Sicherheitsimplementierung. WebSphere eXtreme Scale wurde erfolgreich in die folgenden Komponenten integriert:

- Lightweight Directory Access Protocol (LDAP)
- Kerberos
- ObjectGrid-Sicherheit
- Tivoli Access Manager
- Java Authentication and Authorization Service (JAAS)

eXtreme Scale verwendet den Sicherheitsprovider für die folgenden Tasks:

- Authentifizierung von Clients bei Servern,
- Berechtigung von Clients für den Zugriff auf bestimmte Artefakte von eXtreme Scale oder Festlegung der Verwendung von eXtreme-Scale-Artefakten.

eXtreme Scale hat die folgenden Typen von Berechtigungen:

Map-Berechtigung
: Clients oder Gruppen können für die Durchführung von Einfüge-, Lese-, Aktualisierungs- oder Löschoperationen in Maps berechtigt werden.

ObjectGrid-Berechtigung
: Clients oder Gruppen können für die Ausführung von Objekt- oder Entitätsabfragen in ObjectGrids berechtigt werden.

DataGrid-Agentenberechtigung
: Clients oder Gruppen können für die Implementierung von DataGrid-Agenten in ein ObjectGrid berechtigt werden.

Serverseitige Map-Berechtigung
: Clients oder Gruppen können für die Replikation einer Server-Map auf Clientseite oder die Erstellung eins dynamischen Index für die Server-Map berechtigt werden.

Verwaltungsberechtigung
: Clients oder Gruppen können für die Ausführung von Verwaltungs-Tasks berechtigt werden.

Anmerkung: Wenn Sie die Sicherheit für Ihr Back-End bereits aktiviert haben, müssen Sie beachten, dass diese Sicherheitseinstellungen für den Schutz Ihrer Daten nicht mehr ausreichen. Die Sicherheitseinstellungen Ihrer Datenbank oder eines anderen Datenspeichers werden in keiner Weise auf Ihren Cache übertragen. Sie müssen die Daten, die jetzt zwischengespeichert werden, mit dem Sicherheitsmechanismus von eXtreme Scale schützen, der Authentifizierung, Berechtigung und Sicherheit auf Transportebene umfasst.

Einschränkung: Verwenden Sie JDK/JRE 1.6 und höher nicht, wenn Sie die SSL-Sicherheit auf Transportschicht in einer eigenständigen Version von WebSphere eXtreme Scale 7.1 (oder 7.0) einsetzen. JDK/JRE 1.6 und höher unterstützen die Anwendungsprogrammierschnittstellen von WXS 7.1 nicht. Verwenden Sie JDK/JRE 1.5 oder ältere Versionen für Konfigurationen, die die SSL-Transportsicherheit für

eigenständige Installationen von eXtreme Scale erfordern. Dies gilt nur, wenn die SSL-Sicherheit in eigenständigen Konfigurationen von eXtreme Scale verwendet wird. JDK/JRE wird für Konfigurationen ohne SSL-Transportsicherheit unterstützt.

Integration der Sicherheit mit WebSphere Application Server

WebSphere eXtreme Scale stellt mehrere Sicherheitsfeatures für die Integration in die Sicherheitsstruktur von WebSphere Application Server bereit.

Integration der Authentifizierung

Wenn eXtreme-Scale-Clients und -Server in WebSphere Application Server und in derselben Sicherheitsdomäne ausgeführt werden, können Sie die Sicherheitsinfrastruktur von WebSphere Application Server verwenden, um die Berechtigungsnachweise für die Clientauthentifizierung an den eXtreme-Scale-Server weiterzugeben. Versucht ein Servlet beispielsweise als eXtreme-Scale-Client eine Verbindung zu einem eXtreme-Scale-Server in derselben Sicherheitsdomäne herzustellen und ist das Servlet bereits authentifiziert, kann das Authentifizierungstoken von Client (Servlet) an den Server weitergegeben und anschließend die Sicherheitsinfrastruktur von WebSphere Application Server verwendet werden, um das Authentifizierungstoken an die Clientberechtigungsnachweise zurückzugeben.

Integration der Sicherheit für verteilte Umgebungen mit WebSphere Application Server

Für das verteilte ObjectGrid-Modell kann die Integration der Sicherheit über die folgenden Klassen vorgenommen werden:

`com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator`

`com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator`

`com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredential`

Weitere Informationen finden Sie im Abschnitt „Anwendungsclientauthentifizierung” auf Seite 364. Das folgende Beispiel veranschaulicht, wie die Klasse "WSTokenCredentialGenerator" verwendet wird:

```
/**
 * Verbindung zum ObjectGrid-Server herstellen
 */
protected ClientClusterContext connect() throws ConnectException {
  ClientSecurityConfiguration csConfig = ClientSecurityConfigurationFactory
  .getClientSecurityConfiguration(proFile);

  CredentialGenerator gen = getWSCredGen();

  csConfig.setCredentialGenerator(gen);

  return objectGridManager.connect(csConfig, null);
}

/**
 * WSTokenCredentialGenerator abrufen
 *
private CredentialGenerator getWSCredGen() {
    WSTokenCredentialGenerator gen = new WSTokenCredentialGenerator(
        WSTokenCredentialGenerator.RUN_AS_SUBJECT);
    return gen;
}
```

Verwenden Sie serverseitig den WSTokenAuthentication-Authentifikator, um das WSTokenCredential-Objekt zu authentifizieren.

Integration der Sicherheit für lokale Umgebungen mit WebSphere Application Server

Für das lokale ObjectGrid-Modell kann die Integration der Sicherheit über die folgenden beiden Klassen vorgenommen werden:

- com.ibm.websphere.objectgrid.security.plugins.builtins.WSSubjectSourceImpl
- com.ibm.websphere.objectgrid.security.plugins.builtins.WSSubjectValidationImpl

Weitere Informationen zu diesen Klassen finden Sie in der Beschreibung der lokalen Sicherheit im *Programmierhandbuch*. Sie können die Klasse "WSSubjectSourceImpl" als SubjectSource-Plug-in und die Klasse "WSSubjectValidationImpl" als SubjectValidation-Plug-in konfigurieren.

Sichere eXtreme-Scale-Server starten und stoppen

Server müssen für die Implementierungsumgebung häufig sicher sein, und dies erfordert eine spezielle Konfiguration für das Starten und Stoppen der Server.

Sicheren Server in einer Java-SE-Umgebung starten

Sie können einen Katalogservice oder Containerserver wie folgt starten.

Sicheren eXtreme-Scale-Katalogservice starten

Für das Starten eines sicheren eXtreme-Scale-Katalogserviceprozesses sind zwei weitere Sicherheitskonfigurationsdateien erforderlich:

XML-Sicherheitsdeskriptordatei: Die XML-Sicherheitsdeskriptordatei beschreibt die Sicherheitseigenschaften, die für alle Server (einschließlich Katalogservern und Containerservern) gelten. Ein Beispiel für eine solche Eigenschaft ist die Authentifikatorkonfiguration, die die Benutzer-Registry und das Authentifizierungsverfahren darstellt.

Servereigenschaftendatei. Die Servereigenschaftendatei konfiguriert die für den Server spezifischen Sicherheitseigenschaften.

Wenn Sie den Befehl "startOgServer.sh" bzw. "startOgServer.cat" zum Starten eines sicheren eXtreme-Scale-Katalogserviceprozesses verwenden, können Sie "-clusterSecurityFile" oder "-clusterSecurityUrl" verwenden, um die XML-Sicherheitsdeskriptordatei als Dateityp oder URL-Typ festzulegen, und Sie können "–serverProps" verwenden, um die Servereigenschaftendatei zu setzen.

Sicheren eXtreme-Scale-Containerserver starten

Für das Starten eines sicheren eXtreme-Scale-Containerservers ist eine einzige Sicherheitskonfigurationsdatei erforderlich:

- **Servereigenschaftendatei:** Die Servereigenschaftendatei konfiguriert die für den Server spezifischen Sicherheitseigenschaften. Weitere Einzelheiten finden Sie im Abschnitt „Servereigenschaftendatei” auf Seite 188.

Wenn Sie den Befehl "startOgServer.sh" bzw. "startOgServer.cat" zum Starten eines sicheren eXtreme-Scale-Containerservers verwenden, können Sie mit "–serverProps" die Servereigenschaftendatei festlegen. Es gibt weitere Methoden zum Festlegen der Servereigenschaftendatei. Weitere Informationen hierzu finden Sie in der Beschreibung der Servereigenschaftendatei.

Einzelheiten zur Verwendung des Befehls "startOgServer.sh" bzw. "startOgServer-.bat" und der zugehörigen Optionen finden Sie im Abschnitt „Script "startOgServer"" auf Seite 340.

Sicheren eXtreme-Scale-Server stoppen

Für das Stoppen eines sicheren eXtreme-Scale-Katalogserviceprozesses bzw. Containerservers ist eine einzige Sicherheitskonfigurationsdatei erforderlich:

- **Clienteigenschaftendatei:** Die Clienteigenschaftendatei kann zum Konfigurieren der Clientsicherheitseigenschaften verwendet werden. Die Clientsicherheitseigenschaften sind erforderlich, damit ein Client eine Verbindung zu einem sicheren Server herstellen kann. Weitere Einzelheiten finden Sie im Abschnitt „Clienteigenschaftendatei" auf Seite 207.

Wenn Sie den Befehl "stopOgServer.sh" bzw. "stopOgServer.cat" zum Stoppen eines sicheren eXtreme-Scale-Katalogserviceprozesses oder -Containerservers verwenden, können Sie mit "-clientSecurityFile" die Sicherheitseigenschaften des Clients festlegen.

Einzelheiten zur Verwendung des Befehls "stopOgServer.sh" bzw. "stopOgServer-.cat" und der zugehörigen Optionen finden Sie im Abschnitt „Script "stopOgServer"" auf Seite 345.

Sicheren Server in WebSphere Application Server starten

Das Starten eines sicheren ObjectGrid-Servers in WebSphere Application Server verläuft ähnlich wie das Starten eines nicht sicheren ObjectGrid-Servers, abgesehen davon, dass Sie die Sicherheitskonfigurationsdateien übergeben müssen. Anstatt wie in der J2SE-Umgebung das Argument "-[EIGENSCHAFTENDATEI]" (z. B. -serverProps) im Befehl zu übergeben, verwenden Sie das Argument "-D[EIGENSCHAFTENDATEI] in den generischen JVM-Argumenten.

Sicheren Katalogservice in WebSphere Application Server starten

Ein Katalogserver enthält zwei verschiedene Stufen von Sicherheitsinformationen:

- `-Dobjectgrid.cluster.security.xml.url`: Gibt die Datei "objectGridSecurity.xml" an, die die Sicherheitseigenschaften beschreibt, die für alle Server (einschließlich Katalogservern und Containerservern) gelten. Ein Beispiel ist die Authentifikatorkonfiguration, die die Benutzer-Registry und das Authentifizierungsverfahren darstellt. Der Dateiname für diese Eigenschaft muss im URL-Format angegeben werden, z. B. "file:///tmp/og/objectGridSecurity.xml".
- `-Dobjectgrid.server.props`: Gibt die Servereigenschaftendatei an, die die serverspezifischen Sicherheitseigenschaften enthält. Der für diese Eigenschaft angegebene Dateiname kann im herkömmlichen Dateipfadformat angegeben werden, z. B. "c:/tmp/og/catalogserver.props". Die Verwendung von "-Dobjectgrid.security-.server.props" ist veraltet, aber Sie können diese Option für die Abwärtskompatibilität weiterhin verwenden.

Zum Starten eines sicheren Katalogservice in WebSphere Application Server folgen Sie den Anweisungen unter "Integriert in WebSphere Application Server" im Abschnitt „Grid-Sicherheit" auf Seite 362.

Als Nächstes definieren Sie die Sicherheitseigenschaft in den generischen JVM-Argumenten des Prozesses:

```
-Dobjectgrid.cluster.security.xml.url=file:///tmp/og/
objectGridSecurity.xml-Dobjectgrid.server.props=/tmp/og/
catalog.server.props
```

Im Folgenden wird beschrieben, wie Sie die Eigenschaft den generischen JVM-Argumenten hinzufügen:

- Erweitern Sie den Eintrag "Systemverwaltung" in der Task-Ansicht auf der linken Seite.
- Klicken Sie auf den Prozess von WebSphere Application Server, in dem der Katalogservice implementiert ist, z. B. "Deployment Manager".
- Erweitern Sie auf der rechten Seite den Eintrag "Java- und Prozessverwaltung" unter "Serverinfrastruktur".
- Klicken Sie auf "Prozessdefinition".
- Klicken Sie auf "Java Virtual Machine" unter "Weitere Eigenschaften".
- Geben Sie die Eigenschaften im Textfeld "Generische JVM-Argumente" ein.

Sicheren Containerserver in WebSphere Application Server starten

Ein Containerserver erhält bei der Verbindungsherstellung zum Katalogserver alle Sicherheitskonfigurationen, die in der Datei "objectGridSecurity.xml" konfiguriert sind, wie z. B. die Authentifikatorkonfiguration oder das Zeitlimit für Anmeldesitzungen. Außerdem muss ein Containerserver seine eigenen serverspezifischen Sicherheitseigenschaften in der Eigenschaft "-Dobjectgrid.server.props" konfigurieren.

Die Eigenschaft "-Dobjectgrid.server.props" wird an Stelle der Eigenschaft "-Dobjectgrid.security.server.props" verwendet, weil in diese Eigenschaftendatei auch andere nicht sicherheitsrelevante Eigenschaften aufgenommen werden. Der Dateiname für diese Eigenschaft kann im herkömmlichen Dateipfadformat angegeben werden, z. B. `c:/tmp/og/server.props`.

Führen Sie dieselben Schritte wie zuvor aus, um die Sicherheitseigenschaft den generischen JVM-Argumenten hinzuzufügen.

XML-Sicherheitsdeskriptordatei

Verwenden Sie eine ObjectGrid-XML-Sicherheitsdeskriptordatei, um eine Implementierungstopologie von eXtreme Scale mit aktivierter Sicherheit zu konfigurieren. Die folgenden XML-Musterdateien beschreiben verschiedene Konfigurationen.

Jedes Element und Attribut der XML-Datei wird in der folgenden Liste beschrieben. Verwenden Sie die Beispiele, um sich mit der Verwendung dieser Elemente und Attribute für die Konfiguration der Umgebung vertraut zu machen.

Element "securityConfig"

Das Element "securityConfig" ist das Ausgangselement der ObjectGrid-XML-Sicherheitsdatei. Dieses Element konfiguriert den Namespace der Datei und die Schemaposition. Das Schema wird in der Datei `objectGridSecurity.xsd` definiert.

- Anzahl der Vorkommen: 1
- Untergeordnete Elemente: security

Element "security"

Verwenden Sie das Element "security", um die Sicherheit eines ObjectGrids zu definieren.

- Anzahl der Vorkommen: 1
- Untergeordnete Elemente: authenticator, adminAuthorization und systemCredentialGenerator

Attribute

securityEnabled
Aktiviert die Sicherheit für das Grid, wenn es den Wert "true" hat. Der Standardwert ist "false". Wenn das Attribut den Wert "false" hat, ist die Grid-weite Sicherheit inaktiviert. Weitere Informationen finden Sie im Abschnitt „Grid-Sicherheit" auf Seite 362. (Optional)

singleSignOnEnabled
Wenn Sie das Attribut auf "true" setzen, kann ein Client eine Verbindung zu jedem Server herstellen, nachdem er bei einem der Server authentifiziert wurde. Hat das Attribut den Wert "false", muss ein Client sich bei jedem Server authentifizieren, bevor er eine Verbindung herstellen kann. Der Standardwert ist "false". (Optional)

loginSessionExpirationTime
Gibt die Verfallszeit der Anmeldesitzung in Sekunden an. Wenn die Anmeldesitzung abläuft, muss sich der Client erneut authentifizieren. (Optional)

adminAuthorizationEnabled
Aktiviert die Verwaltungsberechtigung. Wenn dieses Attribut den Wert "true" hat, müssen alle Verwaltungs-Tasks berechtigt werden. Der verwendete Berechtigungsmechanismus wird mit dem Wert des Attributs "adminAuthorizationMechanism" angegeben. Der Standardwert ist "false". (Optional)

adminAuthorizationMechanism
Gibt an, welcher Berechtigungsmechanismus zu verwenden ist. WebSphere eXtreme Scale unterstützt zwei Berechtigungsmechanismen: Java Authentication and Authorization Service (JAAS) und angepasste Berechtigung. Der Berechtigungsmechanismus JAAS verwendet den richtlinienbasierten JAAS-Standardansatz. Wenn Sie JAAS als Berechtigungsmechanismus festlegen möchten, setzen Sie das Attribut auf AUTHORIZATION_MECHANISM_JAAS. Der angepasste Berechtigungsmechanismus verwendet eine vom Benutzer integrierte AdminAuthorization-Implementierung. Wenn Sie einen angepassten Berechtigungsmechanismus angeben möchten, setzen Sie das Attribut auf AUTHORIZATION_MECHANISM_CUSTOM. Weitere Informationen zur Verwendung dieser beiden Mechanismen finden Sie im Abschnitt „Anwendungsclientberechtigung" auf Seite 366. (Optional)

Die folgende Datei `security.xml` ist eine Musterkonfiguration zum Aktivieren der Sicherheit eines eXtreme-Scale-Grids:

security.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<securityConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://ibm.com/ws/objectgrid/config/security ../objectGridSecurity.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config/security">

    <security securityEnabled="true" singleSignOnEnabled="true"
        loginSessionExpirationTime="20"
        adminAuthorizationEnabled="true"
        adminAuthorizationMechanism="AUTHORIZATION_MECHANISM_JAAS" >
```

```
  <authenticator className ="com.ibm.websphere.objectgrid.security.plugins.
builtins.WSTokenAuthenticator">
  </authenticator>

  <systemCredentialGenerator className ="com.ibm.websphere.objectgrid.security.
plugins.builtins.WSTokenCredentialGenerator">
      <property name="properties" type="java.lang.String" value="runAs"
  description="Using runAs subject" />
  </systemCredentialGenerator>

</security></securityConfig>
```

Element "authenticator"

Authentifiziert Clients bei eXtreme-Scale-Servern im Grid. Die Klasse, die mit dem Attribut "className" angegeben wird, muss die Schnittstelle "com.ibm.websphere-.objectgrid.security.plugins.Authenticator" implementieren. Der Authentifikator kann Eigenschaften verwenden, um Methoden in der Klasse aufzurufen, die mit dem Attribut "className" angegeben wird. Weitere Informationen zur Verwendung von Eigenschaften finden Sie in der Beschreibung des Elements "property".

In der vorherigen Beispieldatei `security.xml` wurde die Klasse "com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator" als Authentifikator angegeben. Diese Klasse implementiert die Schnittstelle "com.ibm-.websphere.objectgrid.security.plugins.Authenticator".

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: property

Attribute

className
Gibt eine Klasse an, die die Schnittstelle "com.ibm.websphere.objectgrid.security.plugins.Authenticator" implementiert. Verwenden Sie diese Klasse, um Clients bei Servern im eXtreme-Scale_Grid zu authentifizieren. (Erforderlich)

Element "adminAuthorization"

Verwenden Sie das Element "adminAuthorization", um den Verwaltungszugriff auf das Grid zu konfigurieren.

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: property

Attribute

className
Gibt eine Klasse an, die die Schnittstelle "com.ibm.websphere.objectgrid.security.plugins.AdminAuthorization" implementiert. (Erforderlich)

Element "systemCredentialGenerator"

Verwenden Sie ein Element "systemCredentialGenerator", um einen Generator für Systemberechtigungsnachweise zu konfigurieren. Dieses Element gilt nur für eine dynamische Umgebung. Im dynamischen Konfigurationsmodell stellt der dynamische Containerserver als eXtreme-Scale-Client eine Verbindung zum Katalogserver her, und der Katalogserver kann ebenfalls als Client eine Verbindung zum eXtreme-Scale-Containerserver herstellen. Dieser Generator für Systemberechtigungsnachweise wird für die Darstellung einer Factory für die Systemberechtigungsnachweise verwendet.

- Anzahl der Vorkommen: 0 oder 1

- Untergeordnetes Element: property

Attribute

className
Gibt eine Klasse an, die die Schnittstelle "com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator" implementiert. (Erforderlich)

Ein Beispiel zur Verwendung des Elements "systemCredentialGenerator" entnehmen Sie der vorherigen Datei `security.xml`. In diesem Beispiel ist com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator der Generator für Systemberechtigungsnachweise, der das RunAs-Subject-Objekt aus dem Thread abruft.

Element "property"

Ruft set-Methoden in den Klassen "authenticator" und "adminAuthorization" auf. Der Name der Eigenschaft entspricht einer set-Methode im Attribut "className" des Elements "authenticator" bzw. "adminAuthorization".

- Anzahl der Vorkommen: 0 oder mehr
- Untergeordnetes Element: property

Attribute

name
Der Name der Eigenschaft. Der Wert, der diesem Attribut zugeordnet wird, muss einer set-Methode in der Klasse entsprechen, die mit dem Attribut "className" der übergeordneten Bean angegeben wird. Wenn das Attribut "className" der Bean beispielsweise auf "com.ibm.MyPlugin" gesetzt ist und der Name der angegebenen Eigenschaft "size" lautet, muss die Klasse "com.ibm.MyPlugin" eine Methode "setSize" haben. (Erforderlich)

type
Gibt den Typ der Eigenschaft an. Der Typ des Parameters, der an die mit dem Attribut "name" angegebene set-Methode übergeben wird. Die gültigen Werte sind primitive Java-Typen, ihre java.lang-Pendants und java.lang.String. Die Attribute "name" und "type" müssen einer Methodensignatur im Attribut "className" der Bean entsprechen. Wenn der Name beispielsweise "size" und der Typ "int" ist, muss eine Methode "setSize(int)" in der Klasse vorhanden sein, die mit dem Attribut "className" für die Bean angegeben wurde. (Erforderlich)

value
Gibt den Wert der Eigenschaft an. Dieser Wert wird in den mit dem Attribut "type" angegebenen Typ konvertiert und anschließend als Parameter in dem Aufruf der set-Methode verwendet, die mit den Attributen "name" und "type" angegeben wurde. Der Wert dieses Attributs wird nicht validiert. Der Plug-in-Implementierer muss sicherstellen, dass der übergebene Wert gültig ist. (Erforderlich)

description
Gibt eine Beschreibung der Eigenschaft an. (Optional)

Weitere Informationen finden Sie im Abschnitt „Datei `objectGridSecurity.xsd`" auf Seite 383.

Datei objectGridSecurity.xsd

Verwenden Sie das folgende ObjectGrid-XML-Sicherheitsschema, um die Sicherheit für eine eXtreme-Scale-Implementierung zu aktivieren.

Im Abschnitt „XML-Sicherheitsdeskriptordatei" auf Seite 379 finden Sie Beschreibungen der Elemente und Attribute, die in der Datei objectGridSecurity.xsd definiert sind.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:cc="http://ibm.com/ws/objectgrid/config/security"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://ibm.com/ws/objectgrid/config/security"
  elementFormDefault="qualified">

 <xsd:element name="securityConfig">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="security" type="cc:security" />
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>

 <xsd:complexType name="security">
  <xsd:sequence>
   <xsd:element name="authenticator" type="cc:bean" minOccurs="0"
    maxOccurs="1" />
   <xsd:element name="adminAuthorization" type="cc:bean" minOccurs="0"
    maxOccurs="1" />
   <xsd:element name="systemCredentialGenerator" type="cc:bean" minOccurs="0"
    maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="securityEnabled" type="xsd:boolean" use="optional" />
  <xsd:attribute name="singleSignOnEnabled" type="xsd:boolean" use="optional"/>
  <xsd:attribute name="loginSessionExpirationTime" type="xsd:int" use="optional"/>
  <xsd:attribute name="adminAuthorizationMechanism" type="cc:adminAuthorizationMechanism"
   use="optional"/>
  <xsd:attribute name="adminAuthorizationEnabled" type="xsd:boolean" use="optional" />
 </xsd:complexType>

<xsd:complexType name="bean">
  <xsd:sequence>
   <xsd:element name="property" type="cc:property" maxOccurs="unbounded" minOccurs="0" />
  </xsd:sequence>
  <xsd:attribute name="className" type="xsd:string" use="required"/>
 </xsd:complexType>

<xsd:complexType name="property">
  <xsd:attribute name="name" type="xsd:string" use="required" />
<xsd:attribute name="value" type="xsd:string" use="required" />
  <xsd:attribute name="type" type="cc:propertyType" use="required" />
  <xsd:attribute name="description" type="xsd:string" use="optional" />
 </xsd:complexType>

<xsd:simpleType name="propertyType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="java.lang.Boolean"/>
   <xsd:enumeration value="boolean"/>
   <xsd:enumeration value="java.lang.String"/>
   <xsd:enumeration value="java.lang.Integer"/>
   <xsd:enumeration value="int"/>
   <xsd:enumeration value="java.lang.Double"/>
   <xsd:enumeration value="double"/>
   <xsd:enumeration value="java.lang.Byte"/>
   <xsd:enumeration value="byte"/>
   <xsd:enumeration value="java.lang.Short"/>
   <xsd:enumeration value="short"/>
   <xsd:enumeration value="java.lang.Long"/>
   <xsd:enumeration value="long"/>
   <xsd:enumeration value="java.lang.Float"/>
   <xsd:enumeration value="float"/>
   <xsd:enumeration value="java.lang.Character"/>
   <xsd:enumeration value="char"/>
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:simpleType name="adminAuthorizationMechanism">
```

```
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="AUTHORIZATION_MECHANISM_JAAS" />
   <xsd:enumeration value="AUTHORIZATION_MECHANISM_CUSTOM"/>
  </xsd:restriction>
 </xsd:simpleType>

</xsd:schema>
```

Kapitel 9. Implementierungsumgebung überwachen

Sie können Anwendungsprogrammierschnittstellen (API, Application Programming Interface), MBeans, Protokolle und Dienstprogramme verwenden, um die Leistung Ihrer Anwendungsumgebung zu überwachen.

Übersicht über Statistiken

Statistiken in WebSphere eXtreme Scale werden über eine interne Statistikstruktur erstellt. Die API "StatsAccessor", die PMI-Module (Performance Monitoring Infrastructure) und die MBean-API werden aus der internen Struktur erstellt.

Die folgende Abbildung zeigt eine allgemeine Konfiguration von Statistiken für eXtreme Scale.

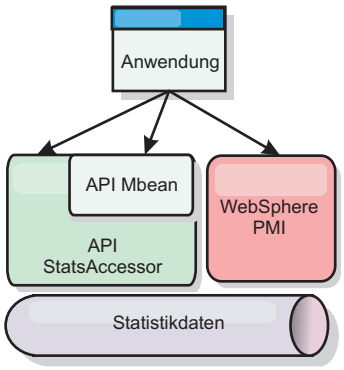


Abbildung 26. Übersicht über Statistiken

Jede dieser APIs bietet eine Sicht auf die Statistikstruktur, wird aber aus jeweils anderen Gründen verwendet:

- **Statistik-API**: Mit der Statistik-API können Entwickler direkt auf Statistiken zugreifen, was flexible und anpassbare Lösungen für die Integration von Statistiken wie angepasste MBeans oder Protokollierung ermöglicht.
- **API MBean**: Die API "MBean" ist ein spezifikationsbasierter Mechanismus für die Überwachung. Die API "MBean" verwendet die Statistik-API und wird lokal in der JVM des Servers ausgeführt. Die API- und MBean-Strukturen sind so konzipiert, dass sie problemlos in die Dienstprogramme anderer Anbieter integriert werden können. Verwenden Sie die API "MBean", wenn Sie mit einem verteilten ObjectGrid arbeiten.
- **PMI-Module (Performance Monitoring Infrastructure) von WebSphere Application Server:** Verwenden Sie PMI, wenn Sie WebSphere eXtreme Scale in WebSphere Application Server ausführen. Diese Module liefern eine Sicht der internen Statistikstruktur.

Statistik-API

Wie bei einer Baumstruktur-Map gibt es einen entsprechenden Pfad und einen Schlüssel, um ein bestimmtes Modul abzurufen, bzw. in diesem Fall eine Differenzierungs- oder Aggregationsstufe. Angenommen, es gibt bereits einen Stammknoten in der Baumstruktur und die Statistiken werden für eine Map mit dem Namen

"payroll" erfasst, die zu einem ObjectGrid mit dem Namen "accounting" gehört. Um beispielsweise auf das Modul für die Aggregations- bzw. Differenzierungsstufe einer Map zuzugreifen, können Sie einen Zeichenfolgebereich (String[]) der Pfade übergeben. In diesem Fall würde dieser "String[] {root, "accounting", "payroll"}" lauten, da jede Zeichenfolge den Pfad des Knotens darstellt. Der Vorteil dieser Struktur ist der, dass ein Benutzer den Bereich für jeden Knoten im Pfad angeben und die Aggregationsstufe für diesen Knoten abrufen kann. Wenn Sie also "String[] {root, "accounting"}" übergeben, erhalten Sie zwar Map-Statistiken, allerdings für das gesamte Grid "accounting." Damit kann der Benutzer sowohl die Typen der zu überwachenden Statistiken als auch die für die Anwendung erforderliche Aggregationsstufe angeben.

PMI-Module von WebSphere Application Server

WebSphere eXtreme Scale enthält Statistikmodule, die mit WebSphere Application Server PMI verwendet werden können. Wenn ein Profil von WebSphere Application Server mit WebSphere eXtreme Scale erweitert wird, integrierten die Erweiterungsscripts die Module von WebSphere eXtreme Scale automatisch in die Konfigurationsdateien von WebSphere Application Server. Mit PMI können Sie Statistikmodule aktivieren und inaktivieren, Statistiken automatisch mit verschiedenen Differenzierungsstufen automatisch zusammenfassen und die Daten über den integrierten Tivoli Performance Viewer selbst in einem Graphen darstellen. Weitere Informationen hierzu finden Sie im Abschnitt „Überwachung mit WebSphere Application Server PMI” auf Seite 394.

Integration von Managed Beans (MBean) in Produkte anderer Anbieter

Die APIs und Managed Beans von eXtreme Scale sind so konzipiert, dass sie problemlos in Überwachungsanwendungen anderer Anbieter integriert werden können. JConsole und MC4J sind zwei Beispiele für schlanke JMX-Konsolen (Java Management Extensions), die zum Analysieren von Informationen über eine eXtreme-Scale-Topologie verwendet werden können. Sie können auch die programmgesteuerten APIs verwenden, um Adapterimplementierungen zu schreiben, mit denen Momentaufnahmen der eXtreme-Scale-Leistung erstellt und die eXtreme-Scale-Leistung überwacht werden können. WebSphere eXtreme Scale enthält eine Musterüberwachungsanwendung, die sofort einsatzfähige Funktionen bietet und die als Schablone für das Schreiben erweiterter angepasster Überwachungsdienstprogramme verwendet werden kann.

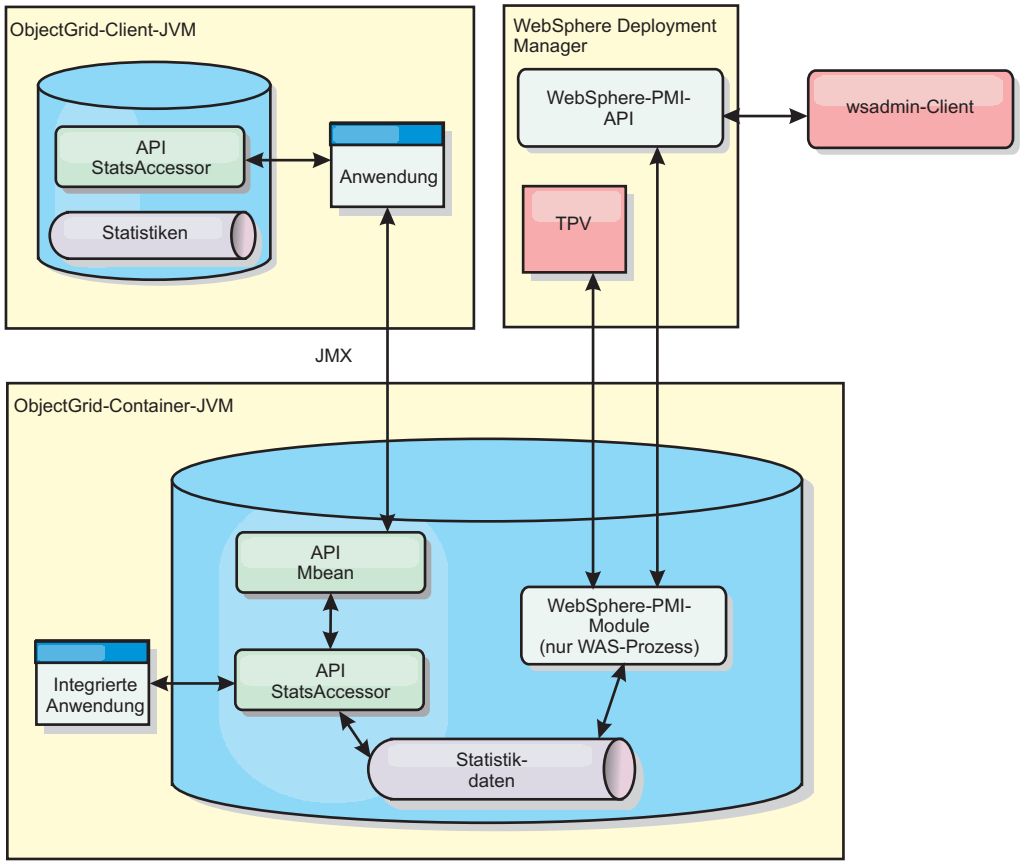


Abbildung 27. Übersicht über MBeans

Weitere Informationen hierzu finden Sie im Abschnitt „Überwachung mit dem Musterdienstprogramm "xsAdmin"" auf Seite 391. Weitere Informationen zur Integration bestimmter Anwendungen anderer Anbieter finden Sie im folgenden Abschnitt:

- eXtreme Scale mit IBM Tivoli Monitoring Agent überwachen
- „eXtreme Scale mit Hyperic HQ überwachen" auf Seite 424
- „eXtreme-Scale-Anwendungen mit CA Wily Introscope überwachen" auf Seite 420

Überwachung mit der Statistik-API

Die Statistik-API ist die direkte Schnittstelle zur internen Statistikstruktur. Statistiken sind standardmäßig inaktiviert, können aber über die Definition einer Schnittstelle "StatsSpec" aktiviert werden. Eine Schnittstelle "StatsSpec" definiert, wie WebSphere eXtreme Scale Statistiken überwachen soll.

Informationen zu diesem Vorgang

Sie können die lokale API "StatsAccessor" verwenden, um Daten abzufragen und auf Statistiken zu jeder ObjectGrid-Instanz zuzugreifen, die in derselben Java Virtual Machine (JVM) wie der aktive Code ausgeführt wird. Weitere Informationen zu den einzelnen Schnittstellen finden Sie in der API-Dokumentation. Verwenden Sie die folgenden Schritte, um die Überwachung der internen Statistikstruktur zu aktivieren.

Vorgehensweise

1. Rufen Sie das StatsAccessor-Objekt ab. Die Schnittstelle "StatsAccessor" folgt dem Singleton-Muster. Abgesehen von Problemen mit dem Klassenladeprogramm sollte deshalb nur eine einzige StatsAccessor-Instanz für jede JVM vorhanden sein. Diese Klasse dient als Hauptschnittstelle für alle lokalen Statistikoperationen. Der folgende Beispielcode veranschaulicht, wie die Klasse "accessor" abgerufen wird. Rufen Sie diese Operation auf, bevor Sie andere ObjectGrid-Aufrufe absetzen.

```
public class LocalClient {
public static void main(String[] args) {

      // Handle für StatsAccessor abrufen
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

   }

}
```

2. Definieren Sie die Schnittstelle "StatsSpec" für das Grid. Definieren Sie diese JVM so, dass alle Statistiken nur auf ObjectGrid-Ebene erfasst werden. Sie müssen sicherstellen, dass eine Anwendung alle Statistiken aktiviert, die möglicherweise erforderlich sind, bevor Sie Transaktionen starten. Im folgenden Beispiel wird die Schnittstelle StatsSpec mit einem statischen Konstantenfeld und einer Spezifikationszeichenfolge definiert. Die Verwendung eines statischen Konstantenfelds ist einfacher, weil das Feld bereits die Spezifikation definiert hat. Wenn Sie jedoch eine Spezifikationszeichenfolge verwenden, können Sie jede erforderliche Kombination von Statistiken aktivieren.

```
public static void main(String[] args) {

      // Handle für StatsAccessor abrufen
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Spezifikation über das statische Feld definieren.
      StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);
      accessor.setStatsSpec(spec);

      // Spezifikation über die Spezifikationszeichenfolge definieren.
      StatsSpec spec = new StatsSpec("og.all=enabled");
      accessor.setStatsSpec(spec);

}
```

3. Senden Sie Transaktionen an das Grid, damit Daten für die Überwachung erfasst werden. Zum Erfassen hilfreicher Daten für Statistiken müssen Sie Transaktionen an das Grid senden. Der folgende Codeauszug fügt einen Datensatz in MapA ein, die in ObjectGridA enthalten ist. Da die Statistiken auf ObjectGrid-Ebene erfasst werden, liefert jede Map im ObjectGrid dieselben Ergebnisse.

```
public static void main(String[] args) {

      // Handle für StatsAccessor abrufen
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Spezifikation über das statische Feld definieren.
      StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);
      accessor.setStatsSpec(spec);

      ObjectGridManager manager =
    ObjectGridmanagerFactory.getObjectGridManager();
      ObjectGrid grid = manager.getObjectGrid("ObjectGridA");
      Session session = grid.getSession();
      Map map = session.getMap("MapA");

      // Einfügenvorgang starten.
```

```
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();
}
```

4. Fragen Sie eine StatsFact mit der API "StatsAccessor" ab. Jedem Statistikpfad wird eine Schnittstelle "StatsFact" zugeordnet. Die Schnittstelle StatsFact ist ein generischer Platzhalter, der verwendet wird, um ein StatsModule-Objekt zu organisieren und aufzunehmen. Bevor Sie auf das eigentliche Statistikmodul zugreifen können, muss das StatsFact-Objekt abgerufen werden.

```
public static void main(String[] args)
{

      // Handle für StatsAccessor abrufen
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Spezifikation über das statische Feld definieren.
      StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);
      accessor.setStatsSpec(spec);

      ObjectGridManager manager =
    ObjectGridManagerFactory.getObjectGridManager();
      ObjectGrid grid = manager.getObjectGrid("ObjectGridA");
      Session session = grid.getSession();
      Map map = session.getMap("MapA");

      // Einfügenvorgang starten.
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();

      // StatsFact abrufen

      StatsFact fact = accessor.getStatsFact(new String[] {"EmployeeGrid"},
    StatsModule.MODULE_TYPE_OBJECT_GRID);

}
```

5. Interagieren Sie mit dem StatsModule-Objekt. Das StatsModule-Objekt ist in der Schnittstelle StatsFact enthalten. Sie können eine Referenz auf das Modul über die Schnittstelle StatsFact anfordern. Da die Schnittstelle StatsFact eine generische Schnittstelle ist, müssen Sie das zurückgegebene Modul in den erwarteten StatsModule-Typ umsetzen. Da diese Task eXtreme-Scale-Statistiken erfasst, wird das zurückgegebene StatsModule-Objekt in den Typ "OGStatsModule" umgesetzt. Nach der Umsetzung des Moduls haben Sie Zugriff auf alle verfügbaren Statistiken.

```
public static void main(String[] args) {

      // Handle für StatsAccessor abrufen
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Spezifikation über das statische Feld definieren.
      StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);
      accessor.setStatsSpec(spec);

      ObjectGridManager manager =
    ObjectGridmanagerFactory.getObjectGridManager();
      ObjectGrid grid = manager.getObjectGrid("ObjectGridA");
      Session session = grid.getSession();
      Map map = session.getMap("MapA");

      // Einfügenvorgang starten.
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();
```

```
        // StatsFact abrufen
        StatsFact fact = accessor.getStatsFact(new String[] {"EmployeeGrid"},
      StatsModule.MODULE_TYPE_OBJECT_GRID);

        // Modul und Zeit abrufen
        OGStatsModule module = (OGStatsModule)fact.getStatsModule();
        ActiveTimeStatistic timeStat =
      module.getTransactionTime("Default", true);
        double time = timeStat.getMeanTime();

  }
```

Statistikmodule

WebSphere eXtreme Scale verwendet ein internes Statistikmodell, um Daten zu überwachen und zu filtern. Dieses Modell ist die grundlegende Struktur, die alle Datensichten verwenden, um Momentaufnahmen von Statistiken zu erfassen. Sie können verschiedene Methoden verwenden, um die Informationen von den Statistikmodulen abzurufen.

Übersicht

Statistiken in WebSphere eXtreme Scale werden überwacht und sind in StatsModules-Komponenten enthalten. Im Statistikmodell sind mehrere Typen von Statistikmodulen enthalten:

OGStatsModule
: Liefert eine Statistik für eine ObjectGrid-Instanz, einschließlich der Transaktionsantwortzeiten.

MapStatsModule
: Liefert eine Statistik für eine einzelne Map, einschließlich der Anzahl an Einträgen und der Trefferrate.

QueryStatsModule
: Liefert eine Statistik zu Abfragen, einschließlich der Planerstellung und der Ausführungszeiten.

AgentStatsModule
: Liefert eine Statistik zu DataGrid-API-Agenten, einschließlich Serialisierungs- und Ausführungszeiten.

HashIndexStatsModule
: Liefert eine Statistik zu den Ausführungszeiten von HashIndex-Abfragen und Wartung.

SessionStatsModule
: Liefert eine Statistik zum Plug-in für den HTTP-Sitzungsmanager.

Einzelheiten zu den Statistikmodulen finden Sie in in der Beschreibung des Pakets "com.ibm.websphere.objectgrid.stats" in der API-Dokumentation.

Statistiken in einer lokalen Umgebung

Das Modell ist wie eine n-stufige Baumstruktur organisiert, die sich aus allen in der vorherigen Liste erwähnten StatsModule-Typen zusammensetzt. Aufgrund dieser Organisationsstruktur wird jeder Knoten in der Baumstruktur durch die Schnittstelle "StatsFact" dargestellt. Die Schnittstelle "StatsFact" kann ein einzelnes Modul oder zu Aggregationszwecken eine Gruppe von Modulen darstellen. Wenn beispielsweise mehrere Blattknoten in der Baumstruktur bestimmte MapStatsMo-

dule-Objekte darstellen, enthält der übergeordnete StatsFact-Knoten dieser Knoten zusammengefasste Statistiken für alle untergeordneten Module. Nach dem Abruf eines StatsFact-Objekts können Sie die Schnittstelle zum Abrufen des entsprechenden StatsModule verwenden.

Wie in einer Baumstruktur-Map verwenden Sie einen entsprechenden Pfad oder einen Schlüssel, um ein bestimmtes StatsFact abzurufen. Der Pfad ist ein String[]-Wert, der sich aus allen Knoten im Pfad zum angeforderten Fakt zusammengesetzt. Beispiel: Sie haben ein ObjectGrid mit dem Namen "ObjectGridA" erstellt, das zwei Maps enthält: MapA und MapB. Der Pfad zum StatsModule für Map A ist `[ObjectGridA, MapA]`. Der Pfad zu den zusammengefassten Statistiken für beide Maps ist `[ObjectGridA]`.

Statistiken in einer verteilten Umgebung

In einer verteilten Umgebung werden die Statistikmodule über einen anderen Pfad abgerufen. Da ein Server mehrere Partitionen enthalten kann, muss die Statistikbaumstruktur die Partitionen überwachen, zu denen die einzelnen Module gehören. Deshalb ist der Suchpfad für ein bestimmtes StatsFact-Objekt anders. Wenn Sie das vorherige Beispiel erweitern und hinzufügen, dass sich die Maps in Partition 1 befinden, lautet der Pfad zum Abrufen des StatsFact-Objekts für Map A `[1, ObjectGridA, MapA]`.

Überwachung mit dem Musterdienstprogramm "xsAdmin"

Mit dem Musterdienstprogramm "xsAdmin" können Sie Textinformationen zu Ihrer WebSphere eXtreme Scale-Topologie formatieren und anzeigen. Das Musterdienstprogramm stellt eine Methode für die Syntaxanalyse und die Erkennung aktuelle Implementierungsdaten bereit und kann als Grundlage für das Schreiben angepasster Dienstprogramme verwendet werden.

Vorbereitende Schritte

Sie müssen WebSphere eXtreme Scale installiert haben.

Informationen zu diesem Vorgang

Sie können das Musterdienstprogramm "xsAdmin" verwenden, um Feedback zum aktuellen Layout und spezifischen Status des Grids, z. B. zum Map-Inhalt, bereitzustellen. In diesem Beispiel besteht das Layout des Grids in dieser Task aus einem einzigen Grid mit dem Namen *ObjectGridA* mit einer definierten Map mit dem Namen *MapA*, die zum MapSet mit dem Namen *MapSetA* gehört. Dieses Beispiel demonstriert, wie Sie alle aktiven Container in einem Grid anzeigen und gefilterte Metriken bezüglich der Map-Größe von *MapA* ausgeben. Zum Anzeigen aller möglichen Befehlsoptionen führen Sie das Dienstprogramm "xsAdmin" ohne Argumente oder mit der Option **-help** aus.

Vorgehensweise

1. Setzen Sie über die Befehlszeile die Umgebungsvariable JAVA_HOME.
   - UNIX `export JAVA_HOME=javaHome`
   - Windows `set JAVA_HOME=javaHome`
2. Navigieren Sie zum Verzeichnis "bin".
   `cd ObjectGrid-Stammverzeichnis/bin`
3. Starten Sie das Dienstprogramm "xsAdmin".

- **Zum Anzeigen der Onlinehilfe führen Sie den folgenden Befehl aus:**

  UNIX

  ```
  xsadmin.sh
  ```

  Windows

  ```
  xsadmin.bat
  ```

  Sehen Sie sich den Abschnitt mit den erforderlichen Argumenten in der Hilfenachricht an, weil Sie nur eine einzige der aufgelisteten Optionen übergeben dürfen, damit das Dienstprogramm funktioniert. Wenn Sie keine Option **-g** oder **-m** angeben, gibt das Dienstprogramm "xsAdmin" Informationen zu jedem Grid in der Topologie aus.
- **Führen Sie den folgenden Befehl aus, um die Statistiken für alle Server zu aktivieren:**

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -setstatsspec ALL=enabled
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -setstatsspec ALL=enabled
  ```
- **Wenn Sie alle Onlinecontainer für ein Grid anzeigen möchten, führen Sie den folgenden Befehl aus:**

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -m MapSetA -containers
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -m MapSetA -containers
  ```

  Alle Containerinformationen werden angezeigt. Es folgt ein Beispiel für die Ausgabe:

  ```
  This administrative utility is provided as a sample only and is not to be
  considered a fully supported component of the WebSphere eXtreme Scale product

  Connecting to Catalog service at localhost:1099


  *** Show all online containers for grid - ObjectGridA & mapset - MapSetA

  Host: 192.168.0.186
  Container: server1_C-0, Server:server1, Zone:DefaultZone
  Partition Shard Type
          0 Primary

  Num containers matching = 1
  Total known containers = 1
  Total known hosts = 1
  ```
- **Führen Sie den folgenden Befehl aus, um die Verbindung zum Katalogservice herzustellen und Informationen zu MapA anzuzeigen:**

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  ```

  Die Größe der angegebenen Map wird angezeigt. Es folgt ein Beispiel für die Ausgabe:

```
This administrative utility is provided as a sample only and is not to be
considered a fully supported component of the WebSphere eXtreme Scale product

Connecting to Catalog service at localhost:1099

****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


*** Listing Maps for server1 ***
 Map Name Partition Map Size Used Bytes (B) Shard Type
 MapA     0         0        0              Primary
```

- **Führen Sie den folgenden Befehl aus, um eine Verbindung zum Katalogservice unter Verwendung eines bestimmten JMX-Ports herzustellen und Informationen zu MapA anzuzeigen:** UNIX

```
xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  -ch CatalogMachine -p 6645
```

Windows

```
xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  -ch CatalogMachine -p 6645
```

Das Musterdienstprogramm "xsAdmin" stellt eine Verbindung zum MBean-Server her, der auf einem Katalogserver ausgeführt wird. Ein Katalogserver kann in einem eigenständigen Prozess, einem Prozess von WebSphere Application Server oder integriert in einem angepassten Anwendungsprozess ausgeführt werden. Verwenden Sie die Option **-ch**, um den Hostnamen des Katalogservice anzugeben, und die Option **-p**, um den Port des Katalogservice anzugeben.

Die Größe der angegebenen Map wird angezeigt. Es folgt ein Beispiel für die Ausgabe:

```
This administrative utility is provided as a sample only and is not to be
considered a fully supported component of the WebSphere eXtreme Scale product

Connecting to Catalog service at CatalogMachine:6645

*****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


*** Listing Maps for server1 ***
Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
Server Total: 0
```

- **Wenn Sie eine Verbindung zu einem Katalogservice in einem Prozess von WebSphere Application Server herzustellen, führen Sie die folgenden Schritte aus:**

  Die Option **-dmgr** ist erforderlich, wenn eine Verbindung zu einem Katalogservice in einem Prozess oder Prozesscluster von WebSphere Application Server hergestellt werden soll. Verwenden Sie die Option **-ch**, um den Hostnamen anzugeben, falls dieser nicht `localhost` ist, und die Option **-p**, um den Bootstrap-Port des Katalogservice zu überschreiben, der den mit BOOTSTRAP_ADDRESS angegebenen Prozess verwendet. Die Option **-p** ist nur erforderlich, wenn BOOTSTRAP_ADDRESS nicht auf den Standardwert von 9809 gesetzt ist.

  Anmerkung: Die eigenständige Version von WebSphere eXtreme Scale kann nicht verwendet werden, um eine Verbindung zu einem Katalogservice in einem Prozess von WebSphere Application Server herzustellen. Verwenden Sie das Script "xsAdmin" im Verzeichnis *WAS-Stammverzeichnis*`/bin`, das verfüg-

bar ist, wenn Sie WebSphere eXtreme Scale in WebSphere Application Server oder WebSphere Application Server Network Deployment installieren.

a. Navigieren Sie zum Verzeichnis "bin" von WebSphere Application Server.

```
cd WAS-Stammverzeichnis/bin
```

b. Starten Sie das Dienstprogramm "xsAdmin" mit dem folgenden Befehl:

UNIX

```
xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
```

Windows

```
xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
```

Die Größe der angegebenen Map wird angezeigt.

```
This administrative utility is provided as a sample only and is not to be
considered a fully supported component of the WebSphere eXtreme Scale product

Connecting to Catalog service at localhost:9809

****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


*** Listing Maps for server1 ***
Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
Server Total: 0
```

Überwachung mit WebSphere Application Server PMI

WebSphere eXtreme Scale unterstützt Performance Monitoring Infrastructure (PMI), wenn Sie mit einem Anwendungsserver von WebSphere Application Server oder WebSphere Extended Deployment arbeiten. PMI erfasst Leistungsdaten zu Laufzeitanwendungen und stellt Schnittstellen bereit, über die externe Anwendungen für die Überwachung von Leistungsdaten unterstützt werden. Sie können die Administrationskonsole oder das Tool "wsadmin" verwenden, um auf Überwachungsdaten zuzugreifen.

Vorbereitende Schritte

Sie können PMI verwenden, um Ihre Umgebung zu überwachen, wenn Sie WebSphere eXtreme Scale in Kombination mit WebSphere Application Server verwenden.

Informationen zu diesem Vorgang

WebSphere eXtreme Scale verwendet das angepasste PMI-Feature von WebSphere Application Server, um eine eigene PMI-Instrumentierung hinzuzufügen. Über diesen Ansatz können Sie WebSphere eXtreme Scale PMI mit der Administrationskonsole oder mit JMX-Schnittstellen (Java Management Extensions) im Tool "wsadmin" aktivieren oder inaktivieren. Außerdem können Sie über die Standard-PMI- und -JMX-Schnittstellen, die von Überwachungsstools wie Tivoli Performance Viewer verwendet werden, auf Statistiken von WebSphere eXtreme Scale zugreifen.

Vorgehensweise

1. Aktivieren Sie PMI in eXtreme Scale. Sie müssen PMI aktivieren, um die PMI-Statistiken anzeigen zu können. Weitere Informationen hierzu finden Sie im Abschnitt „PMI aktivieren” auf Seite 395.

2. Rufen Sie PMI-Statistiken von eXtreme Scale ab. Zeigen Sie die Leistung Ihrer eXtreme-Scale-Anwendungen mit Tivoli Performance Viewer an. Weitere Informationen hierzu finden Sie im Abschnitt „PMI-Statistiken abrufen” auf Seite 397.

Nächste Schritte

Weitere Informationen zum Tool "wsadmin" finden Sie im Abschnitt „Mit dem Tool "wsadmin" auf MBeans zugreifen” auf Seite 360.

PMI aktivieren

Sie können WebSphere Application Server Performance Monitoring Infrastructure (PMI) verwenden, um Statistiken auf jeder Stufe zu aktivieren und zu inaktivieren. So können Sie beispielsweise die Statistik für die Cachetrefferrate einer bestimmten Map aktivieren, und die Statistik für die Eintragsanzahl oder die Statistik für die Loader-Aktualisierungszeiten im Stapelbetrieb inaktivieren. PMI kann über die Administrationskonsole oder mit Scripting aktiviert werden.

Vorbereitende Schritte

Ihr Anwendungsserver muss gestartet sein und eine installierte Anwendung haben, die für eXtreme Scale aktiviert ist. Zum Aktivieren von PMI mit Scripting müssen Sie sich anmelden und das Tool "wsadmin" verwenden können. Weitere Informationen zum Tool "wsadmin" finden Sie im Artikel "Tool 'wsadmin'" im Information Center von WebSphere Application Server.

Informationen zu diesem Vorgang

Verwenden Sie WebSphere Application Server PMI, um einen differenzierten Mechanismus bereitzustellen, mit dem Sie Statistiken auf jeder Stufe aktivieren und inaktivieren können. So können Sie beispielsweise die Statistik für die Cachetreffertrate einer bestimmten Map aktivieren, und die Statistik für die Eintragsanzahl oder die Statistik für die Loader-Aktualisierungszeiten im Stapelbetrieb inaktivieren. In diesem Abschnitt wird beschrieben, wie Sie PMI über die Administrationskonsole und mit wsadmin-Scripts für ObjectGrid aktivieren können.

Vorgehensweise

- **PMI über die Administrationskonsole aktivieren**
  1. Klicken Sie in der Administrationskonsole auf **Überwachung und Optimierung → Performance Monitoring Infrastructure → *Servername***.
  2. Stellen Sie sicher, dass "Performance Monitoring Infrastructure (PMI) aktivieren" ausgewählt ist. Diese Einstellung ist standardmäßig aktiviert. Wenn die Einstellung nicht aktiviert ist, wählen Sie das Kontrollkästchen aus, und starten Sie anschließend den Server erneut.
  3. Klicken Sie auf **Angepasst**. Wählen Sie in der Konfigurationsstruktur das ObjectGrid und das ObjectGrid-Modul "Maps" aus. Aktivieren Sie die Statistik für jedes Modul.

  Die Transaktionstypkategorie für ObjectGrid-Statistiken wird zur Laufzeit erstellt. Die Unterkategorien der ObjectGrid- und Map-Statistiken sind nur auf der Registerkarte **Laufzeit** sichtbar.
- **PMI mit Scripting aktivieren**

1. Öffnen Sie eine Befehlszeile. Navigieren Sie zum Verzeichnis "Installationsstammverzeichnis/bin". Geben Sie "wsadmin" ein, um das Befehlszeilentool "wsadmin" zu starten.
2. Ändern Sie die PMI-Laufzeitkonfiguration für eXtreme Scale. Stellen Sie mit den folgenden Befehlen sicher, dass PMI für den Server aktiviert ist:

   ```
   wsadmin>set s1 [$AdminConfig getid /Cell:ZELLENNAME/Node:KNOTENNAME/
       Server:ANWENDUNGSSERVERNAME/]
   wsadmin>set pmi [$AdminConfig list PMIService $s1]
   wsadmin>$AdminConfig show $pmi.
   ```

   Wenn PMI nicht aktiviert ist, führen Sie die folgenden Befehle aus, um PMI zu aktivieren:

   ```
   wsadmin>$AdminConfig modify $pmi {{enable true}}
   wsadmin>$AdminConfig save
   ```

   Wenn Sie PMI aktivieren müssen, starten Sie den Server erneut.
3. Setzen Sie unter Verwendung der folgenden Befehle Variablen, um die Statistikgruppe in eine angepasste Gruppe zu ändern:

   ```
   wsadmin>set perfName [$AdminControl completeObjectName type=Perf,
   process=ANWENDUNGSSERVERNAME,*]
   wsadmin>set perfOName [$AdminControl makeObjectName $perfName]
   wsadmin>set params [java::new {java.lang.Object[]} 1]
   wsadmin>$params set 0 [java::new java.lang.String custom]
   wsadmin>set sigs [java::new {java.lang.String[]} 1]
   wsadmin>$sigs set 0 java.lang.String
   ```
4. Setzen Sie die Statistikgruppe mit dem folgenden Befehl auf "custom" (Angepasst):

   ```
   wsadmin>$AdminControl invoke_jmx $perfOName setStatisticSet $params $sigs
   ```
5. Setzen Sie mit den folgenden Befehlen Variablen, um die PMI-Statistik "objectGridModule" zu aktivieren:

   ```
   wsadmin>set params [java::new {java.lang.Object[]} 2]
   wsadmin>$params set 0 [java::new java.lang.String objectGridModule=1]
   wsadmin>$params set 1 [java::new java.lang.Boolean false]
   wsadmin>set sigs [java::new {java.lang.String[]} 2]
   wsadmin>$sigs set 0 java.lang.String
   wsadmin>$sigs set 1 java.lang.Boolean
   ```
6. Setzen Sie die Statistikzeichenfolge mit dem folgenden Befehl:

   ```
   wsadmin>set params2 [java::new {java.lang.Object[]} 2]
   wsadmin>$params2 set 0 [java::new java.lang.String mapModule=*]
   wsadmin>$params2 set 1 [java::new java.lang.Boolean false]
   wsadmin>set sigs2 [java::new {java.lang.String[]} 2]
   wsadmin>$sigs2 set 0 java.lang.String
   wsadmin>$sigs2 set 1 java.lang.Boolean
   ```
7. Setzen Sie die Statistikzeichenfolge mit dem folgenden Befehl:

   ```
   wsadmin>$AdminControl invoke_jmx $perfOName setCustomSetString $params2 $sigs2
   ```

Mit diesen Schritten haben Sie PMI für die Laufzeitumgebung von eXtreme Scale aktiviert, aber die PMI-Konfiguration nicht geändert. Wenn Sie den Anwendungsserver erneut starten, gehen die PMI-Einstellungen bis auf die eigentliche Aktivierung von PMI verloren.

Beispiel

Sie können die folgenden Schritte ausführen, um die PMI-Statistiken für die Musteranwendung zu aktivieren:

1. Starten Sie die Anwendung mit der Webadresse `http://Host:Port/ObjectGridSample`, wobei "Host" und "Port" für den Hostnamen und die HTTP-Portnummer des Servers stehen, in dem die Musteranwendung installiert ist.
2. Klicken Sie in der Musteranwendung auf "ObjectGridCreationServlet" und anschließend auf die Aktionsschaltflächen 1, 2, 3, 4 und 5, um Aktionen für das ObjectGrid und die Maps zu generieren. Schließen Sie diese Servlet-Seite noch nicht.
3. Klicken Sie in der Administrationskonsole auf **Überwachung und Optimierung** → **Performance Monitoring Infrastructure** → *Servername*. Klicken Sie auf das Register **Laufzeit**.
4. Klicken Sie auf das Optionsfeld **Angepasst**.
5. Erweitern Sie das ObjectGrid-Modul "Maps" in der Laufzeitstruktur, und klicken Sie anschließend auf den Link "clusterObjectGrid". In der ObjectGrid-Gruppe "Maps" gibt es eine ObjectGrid-Instanz "clusterObjectGrid", und in der Gruppe "clusterObjectGrid" gibt es vier Maps: counters, employees, offices und sites. Die ObjectGrid-Instanz enthält die clusterObjectGrid-Instanz, und diese Instanz hat den Transaktionstyp DEFAULT.
6. Sie können die gewünschten Statistiken aktivieren. Sie können beispielsweise die Statistik für die Anzahl der Einträge in der Map "employees" und die Statistik für die Transaktionsantwortzeiten für den Transaktionstyp DEFAULT aktivieren.

Nächste Schritte

Nach der Aktivierung von PMI können Sie die PMI-Statistiken über die Administrationskonsole oder mit Scripting anzeigen.

PMI-Statistiken abrufen

Wenn Sie PMI-Statistiken abrufen, können Sie sich einen Eindruck über die Leistung Ihrer eXtreme-Scale-Anwendungen verschaffen.

Vorbereitende Schritte

- Aktivieren Sie die Verfolgung von PMI-Statistiken für Ihre Umgebung. Weitere Informationen hierzu finden Sie im Abschnitt „PMI aktivieren” auf Seite 395.
- Bei den Pfadangaben in dieser Task wird davon ausgegangen, dass Sie Statistiken für die Musterandung abrufen, aber Sie können diese Statistiken mit ähnlichen Schritten für jede andere Anwendung verwenden.
- Wenn Sie die Administrationskonsole verwenden, um Statistiken abzurufen, müssen Sie sich an der Administrationskonsole anmelden können. Wenn Sie Scripting verwenden, müssen Sie sich bei wsadmin anmelden können.

Informationen zu diesem Vorgang

Sie können PMI-Statistiken abrufen und diese in Tivoli Performance Viewer anzeigen, indem Sie Schritte in der Administrationskonsole oder mit Scripting ausführen.

- Schritte für die Administrationskonsole
- Schritte für Scripting

Weitere Informationen zu den Statistiken, die abgerufen werden können, finden Sie im Abschnitt „PMI-Module” auf Seite 398.

Vorgehensweise

- Rufen Sie PMI-Statistiken in der Administrationskonsole ab.
  1. Klicken Sie in der Administrationskonsole auf **Überwachung und Optimierung → Performance Viewer → Aktuelle Aktivität**.
  2. Wählen Sie den Server, den Sie mit Tivoli Performance Viewer überwachen möchten, aus, und aktivieren Sie anschließend die Überwachung.
  3. Klicken Sie auf den Server, um die Seite "Performance Viewer" anzuzeigen.
  4. Erweitern Sie die Konfigurationsstruktur. Klicken Sie auf **ObjectGrid-Maps → clusterObjectGrid**, und wählen Sie **employees** aus. Klicken Sie auf **ObjectGrids → clusterObjectGrid**, und wählen Sie **DEFAULT** aus.
  5. Navigieren Sie in der ObjectGrid-Musteranwendung zum Servlet "ObjectGridCreationServlet", klicken Sie auf Schaltfläche 1, und füllen Sie die Maps mit Daten. Sie können die Statistiken im Viewer anzeigen.
- Rufen Sie PMI-Statistiken mit Scripting ab.
  1. Navigieren Sie über eine Befehlszeile zum Verzeichnis `Installationsstammverzeichnis/bin`. Geben Sie `wsadmin` ein, um das Tool "wsadmin" zu starten.
  2. Setzen Sie mit den folgenden Befehlen Variablen für die Umgebung:

     ```
     wsadmin>set perfName [$AdminControl completeObjectName type=Perf,*]
     wsadmin>set perfOName [$AdminControl makeObjectName $perfName]
     wsadmin>set mySrvName [$AdminControl completeObjectName type=Server,
      name=APPLICATION_SERVER_NAME,*]
     ```
  3. Setzen Sie mit den folgenden Befehlen Variablen, um die Statistik "mapModule" abzurufen:

     ```
     wsadmin>set params [java::new {java.lang.Object[]} 3]
     wsadmin>$params set 0 [$AdminControl makeObjectName $mySrvName]
     wsadmin>$params set 1 [java::new java.lang.String mapModule]
     wsadmin>$params set 2 [java::new java.lang.Boolean true]
     wsadmin>set sigs [java::new {java.lang.String[]} 3]
     wsadmin>$sigs set 0 javax.management.ObjectName
     wsadmin>$sigs set 1 java.lang.String
     wsadmin>$sigs set 2 java.lang.Boolean
     ```
  4. Rufen Sie die Statistik "mapModule" mit dem folgenden Befehl ab:

     ```
     wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params $sigs
     ```
  5. Setzen Sie mit den folgenden Befehlen Variablen, um die Statistik "objectGridModule" abzurufen:

     ```
     wsadmin>set params2 [java::new {java.lang.Object[]} 3]
     wsadmin>$params2 set 0 [$AdminControl makeObjectName $mySrvName]
     wsadmin>$params2 set 1 [java::new java.lang.String objectGridModule]
     wsadmin>$params2 set 2 [java::new java.lang.Boolean true]
     wsadmin>set sigs2 [java::new {java.lang.String[]} 3]
     wsadmin>$sigs2 set 0 javax.management.ObjectName
     wsadmin>$sigs2 set 1 java.lang.String
     wsadmin>$sigs2 set 2 java.lang.Boolean
     ```
  6. Rufen Sie die Statistik "objectGridModule" mit dem folgenden Befehl ab:

     ```
     wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params2 $sigs2
     ```

Ergebnisse

Sie können Statistiken in Tivoli Performance Viewer anzeigen.

PMI-Module

Sie können die Leistung Ihrer Anwendungen mit den PMI-Modulen (Performance Monitoring Infrastructure) überwachen.

objectGridModule

Das Modul "objectGridModule" enthält eine Zeitstatistik: Antwortzeit der Transaktion. Eine Transaktion ist wie folgt definiert: Dauer zwischen dem Aufruf der Methode "Session.begin" und dem Aufruf der Methode "Session.commit". Diese Dauer wird als Antwortzeit der Transaktion verfolgt. Das Stammelement ("root") des Moduls "the objectGridModule" dient als Einstiegspunkt für die Statistiken von WebSphere eXtreme Scale. Dieses Stammelement hat ObjectGrids als untergeordnete Elemente, die dieselben Transaktionstypen wie deren untergeordnete Elemente haben. Die Antwortzeitstatistik wird jedem Transaktionstyp zugeordnet.

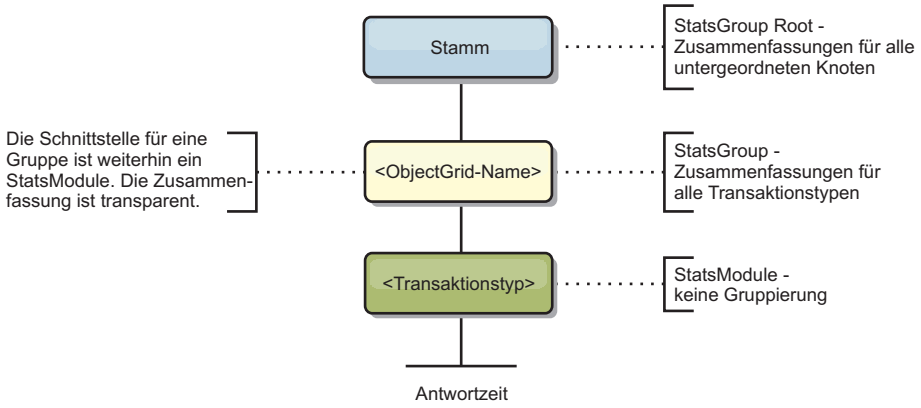


Abbildung 28. Struktur des Moduls "ObjectGridModule"

Die folgende Abbildung zeigt eine ObjectGridModule-Beispielstruktur. In diesem Beispiel sind zwei ObjectGrid-Instanzen im System vorhanden: ObjectGrid A und ObjectGrid B. Die ObjectGrid-Instanz A hat zwei Typen von Transaktionen: A und Standard. Die ObjectGrid-Instanz B hat nur den Transaktionstyp "Standard".

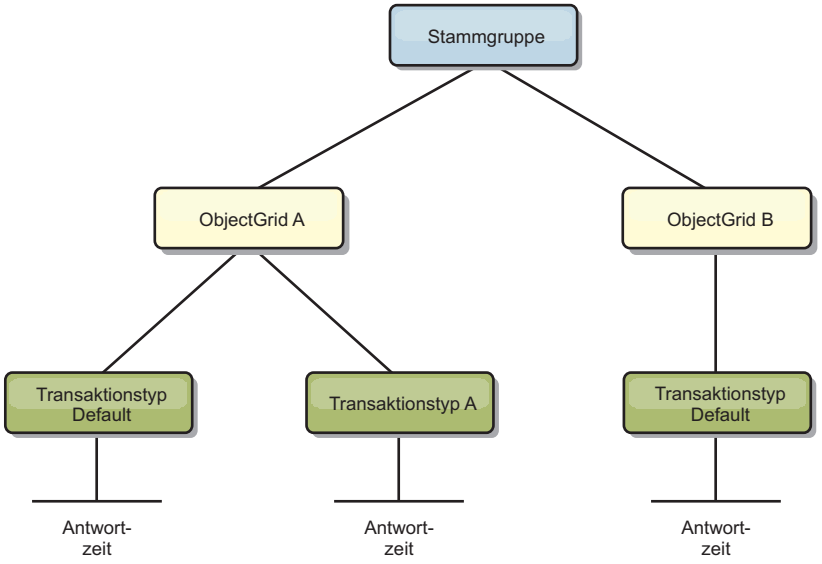


Abbildung 29. Beispielstruktur für das Modul "ObjectGridModule"

Transaktionstypen werden von Anwendungsentwicklern definiert, weil sie wissen, welche Typen von Transaktionen in ihren Anwendungen verwendet werden. Der Transaktionstyp wird mit der folgenden Methode "Session.setTransactionType(String)" gesetzt:

```
/**
* Legt den Transaktionstyp für künftige Transaktionen fest.
*
* Nach dem Aufruf dieser Methode haben alle künftigen Transaktionen denselben
* Typ, bis ein anderer Transaktionstyp festgelegt wird. Wenn Sie keinen
* Transaktionstyp festlegen, wird der Standardtransaktionstyp TRANSACTION_TYPE_DEFAULT
* verwendet.
*
* Transaktionstypen werden hauptsächlich für die Verfolgung statistischer Daten
* verwendet. Benutzer können Typen von Transaktionen, die in einer Anwendung
* ausgeführt werden, vordefinieren. Die Idee ist, Transaktionen mit denselben
* Merkmalen in einer Kategorie (Typ) zusammenzufassen, so dass jeweils eine
* einzige Statistik zur Transaktionsantwortzeit verwendet werden kann, um den
* jeweiligen Transaktionstyp zu verfolgen.
*
* Diese Verfolgung ist hilfreich, wenn Ihre Anwendung unterschiedliche
* Transaktionstypen hat.
* Einige Typen von Transaktionen, wie z. B. Aktualisierungstransaktionen,
* haben eine längere Verarbeitungszeit als andere Transaktionen, wie z. B.
* Lesetransaktionen. Wenn Sie den Transaktionstyp verwenden, können
* unterschiedliche Transaktionen über unterschiedliche Statistiken verfolgt
* werden, so dass die Statistiken hilfreicher sind.
*
* @param tranType Der Transaktionstyp für künftige Transaktionen.
*/
void setTransactionType(String tranType);
```

Im folgenden Beispiel wird der Transaktionstyp auf `updatePrice` gesetzt:

```
// Transaktionstyp auf "updatePrice" setzen.
// Die Zeit zwischen session.begin() und session.commit() wird in der
// Zeitstatistik für "updatePrice" verfolgt.
session.setTransactionType("updatePrice");
session.begin();
map.update(stockId, new Integer(100));
session.commit();
```

Die erste Zeile gibt an, dass der folgende Transaktionstyp "updatePrice" ist. In dem Beispiel ist eine Statistik "updatePrice" in der ObjectGrid-Instanz vorhanden, die dem Session-Objekt entspricht. Mit JMX-Schnittstellen (Java Management Extensions) können Sie die Transaktionsantwortzeit für Transaktionen des Typs "updatePrice" abrufen. Sie können auch die zusammengefasste Statistik für alle Transaktionstypen in der angegebenen ObjectGrid-Instanz abrufen.

mapModule

Das Modul "mapModule" enthält drei Statistiken, die sich auf eXtreme-Scale-Maps beziehen:

- **Map hit rate** - *BoundedRangeStatistic*: Verfolgt die Trefferrate einer Map. Die Trefferrate ist ein variabler Wert zwischen 0 und 100 einschließlich, der den Prozentsatz der Map-Treffer in Relation zu den get-Operationen für die Map darstellt.
- **Number of entries**-*CountStatistic*: Verfolgt die Anzahl der Einträge in der Map.
- **Loader batch update response time**-*TimeStatistic*: Verfolgt die Antwortzeit für Aktualisierungsoperationen im Stapelbetrieb des Loaders.

Das Stammelement ("root") des Moduls "mapModule" dient als Einstiegspunkt für die ObjectGrid-Map-Statistiken. Dieses Stammelement hat Maps als untergeordnete Elemente, die dieselben Transaktionstypen wie deren untergeordnete Elemente haben. Jede Map-Instanz hat die drei aufgelisteten Statistiken. Die Struktur des Moduls "mapModule" ist in der folgenden Abbildung dargestellt:

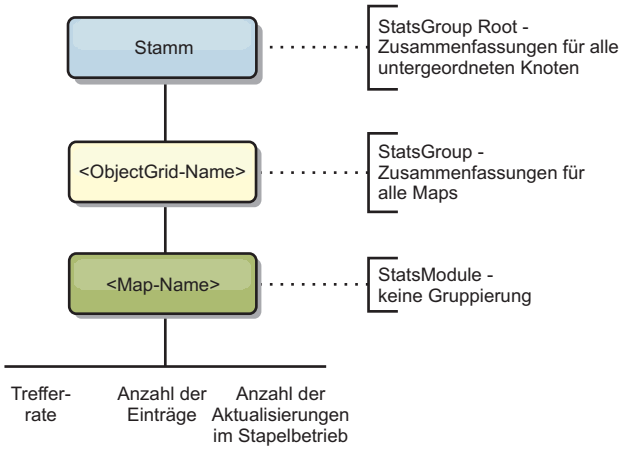
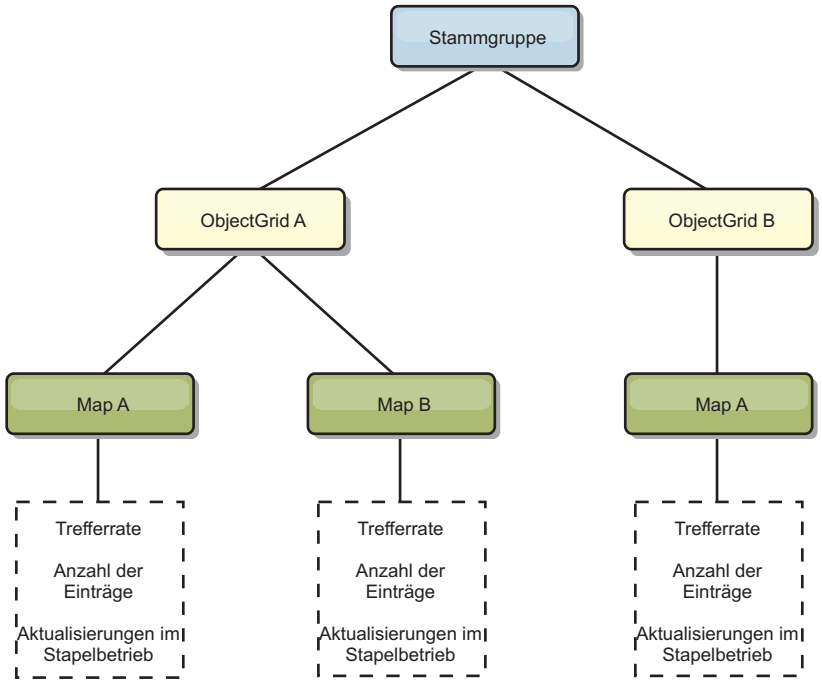


Abbildung 30. Struktur des Moduls "mapModule"

Die folgende Abbildung zeigt eine Beispielstruktur für das Modul "mapModule":

Abbildung 31. Beispielstruktur für das Modul "mapModule"



Modul "hashIndexModule"

Das Modul "hashIndexModule" enthält die folgenden Statistiken, die sich auf Indizes auf Map-Ebene beziehen:

- **Find Count**-*CountStatistic*: Die Anzahl der Aufrufe für die Indexoperation "find".
- **Collision Count**-*CountStatistic*: Die Anzahl der Kollisionen für die Operation "find".
- **Failure Count**-*CountStatistic*: Die Anzahl der Fehler für die Operation "find".
- **Result Count**-*CountStatistic*: Die Anzahl der von der Operation "find" zurückgegebenen Schlüssel.

- **BatchUpdate Count**-*CountStatistic*: Die Anzahl der für diesen Index ausgeführten Aktualisierungen im Stapelbetrieb. Wenn die entsprechende Map geändert wird, wird die Methode "doBatchUpdate()" des Index aufgerufen. Diese Statistik teilt Ihnen mit, wie oft sich der Index ändert bzw. aktualisiert wird.
- **Find Operation Duration Time**-*TimeStatistic*: Ausführungsdauer der Operation "find".

Das Stammelement ("root") des Moduls "hashIndexModule" "root" dient als Einstiegspunkt für die HashIndex-Statistik. Das Stammelement hat ObjectGrids als untergeordnete Elemente, ObjectGrids haben Maps als untergeordnete Elemente, die wiederum HashIndex-Instanzen als untergeordnete Elemente und Blattknoten der Baumstruktur haben. Jede HashIndex-Instanz hat die drei aufgelisteten Statistiken. Die Struktur des Moduls "hashIndexModule" wird in der folgenden Abbildung gezeigt:

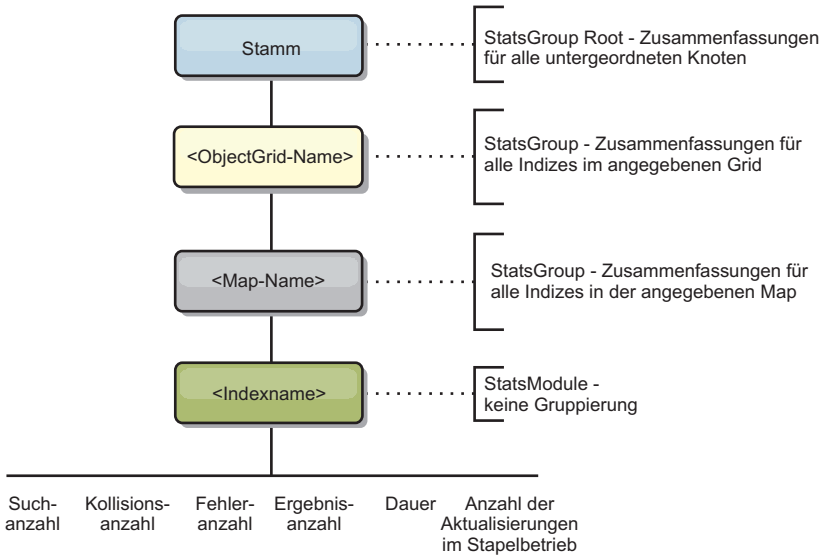


Abbildung 32. Struktur des Moduls "hashIndexModule"

Die folgende Abbildung zeigt eine Beispielstruktur für das Modul "hashIndexModule":

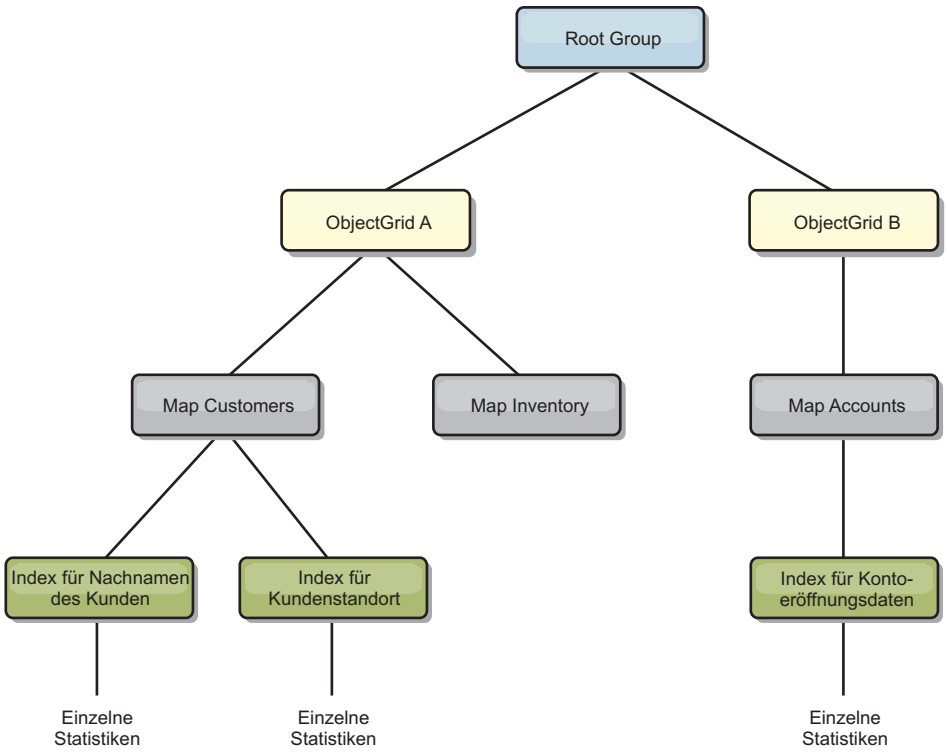


Abbildung 33. Beispielstruktur für das Modul "hashIndexModule"

Modul "agentManagerModule"

Das Modul "agentManagerModule" enthält Statistiken, die sich auf die Agenten auf Map-Ebene beziehen:

- **Reduce Time**: *TimeStatistic* - Die Zeit, die der Agent für die Ausführung der Operation "reduce" benötigt.
- **Total Duration Time**: *TimeStatistic* - Die Gesamtzeit, die der Agent für die Ausführung aller Operationen benötigt.
- **Agent Serialization Time**: *TimeStatistic* - Die Zeit für die Serialisierung des Agenten.
- **Agent Inflation Time**: *TimeStatistic* - Die Zeit, die benötigt wird, um den Agenten im Server zu dekomprimieren.
- **Result Serialization Time**: *TimeStatistic* - Die Zeit für die Serialisierung der Ergebnisse des Agenten.
- **Result Inflation Time**: *TimeStatistic* - Die Zeit für die Dekomprimierung der Ergebnisse des Agenten.
- **Failure Count**: *CountStatistic* - Die Anzahl der Fehler des Agenten.
- **Invocation Count**: *CountStatistic* - Die Anzahl der AgentManager-Aufrufe.
- **Partition Count**: *CountStatistic* - Die Anzahl der Partitionen, an die der Agent gesendet wird.

Das Stammelement ("root" des Moduls "agentManagerModule") dient als Einstiegspunkt für die AgentManager-Statistiken. Dieses Stammelement hat ObjectGrids als untergeordnete Elemente, die ObjectGrids haben Maps als untergeordnete Elemente, die wiederum AgentManager-Instanzen als untergeordnete Elemente und Blattknoten der Baumstruktur haben. Jede AgentManager-Instanz hat die drei aufgelis-

teten Statistiken.

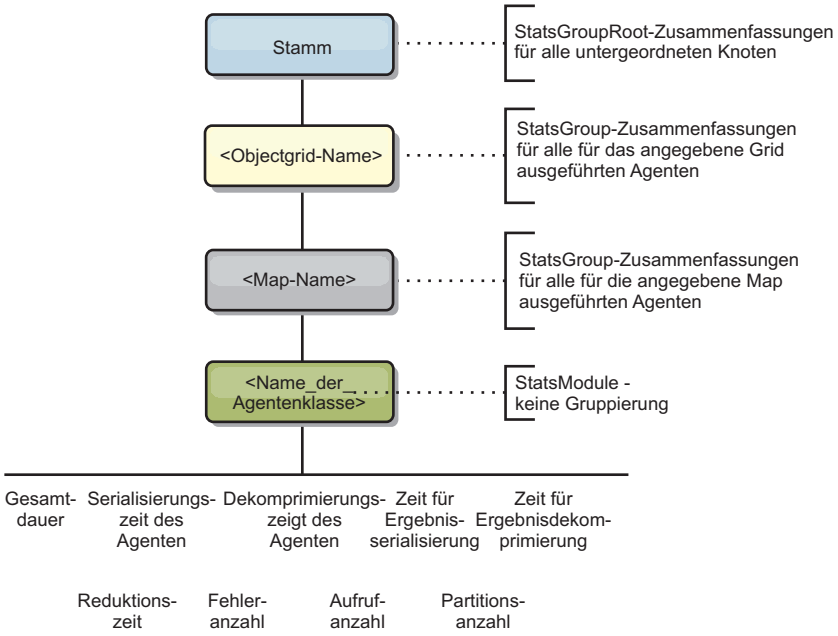


Abbildung 34. Struktur des Moduls "agentManagerModule"

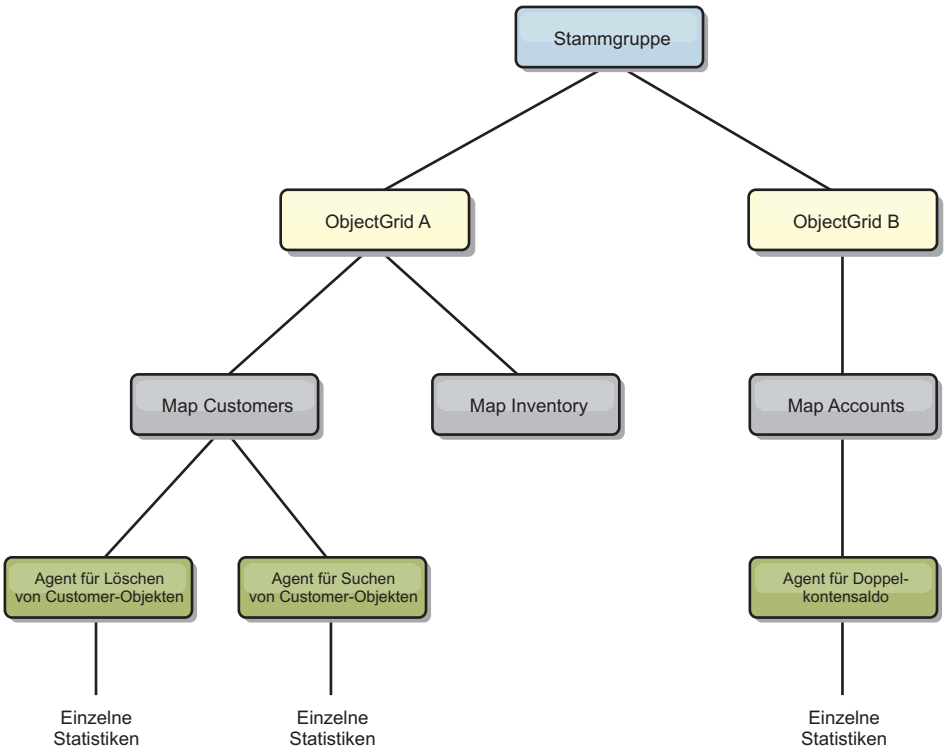


Abbildung 35. Beispielstruktur für das Modul "agentManagerModule"

Modul "queryModule"

Das Modul "queryModule" enthält Statistiken, die sich auf eXtreme-Scale-Abfragen beziehen:

- **Plan Creation Time**: *TimeStatistic* - Die Zeit für die Erstellung des Abfrageplans.

- **Execution Time**: *TimeStatistic* - Die Zeit für die Ausführung der Abfrage.
- **Execution Count**: *CountStatistic* - Die Anzahl der Abfrageläufe.
- **Result Count**: *CountStatistic* - Der Zähler für jede Ergebnismenge jedes Abfragelaufs.
- **FailureCount**: *CountStatistic* - Die Anzahl der Abfragefehler.

Das Stammelement ("root") des Moduls "queryModule" dient als Einstiegspunkt für die Abfragestatistiken. Dieses Stammelement hat ObjectGrids als untergeordnete Elemente, die Query-Objekt als untergeordnete Elemente und Blattknoten der Baumstruktur haben. Jede Query-Instanz hat die drei aufgelisteten Statistiken.

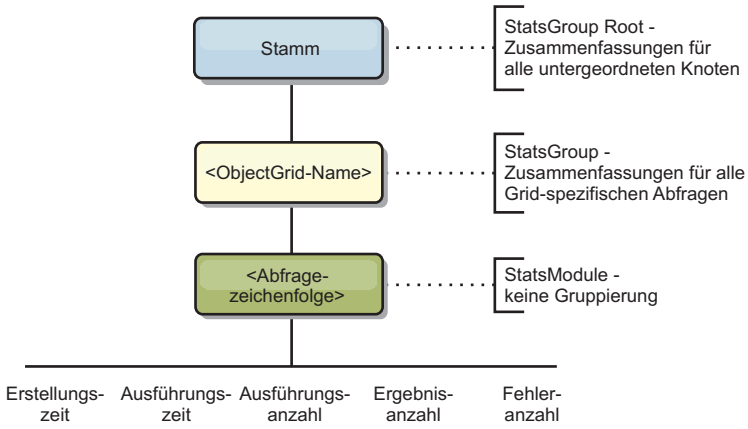


Abbildung 36. Struktur des Moduls "queryModule"

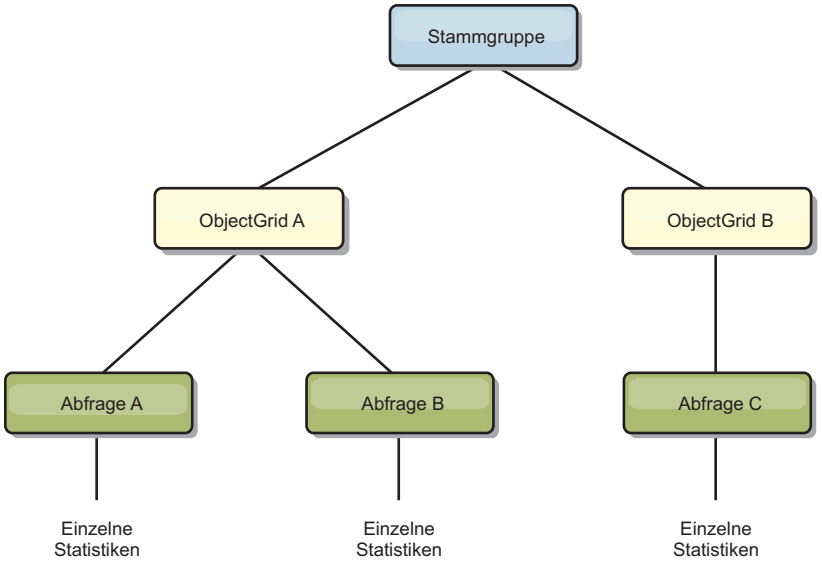


Abbildung 37. Beispielstruktur für das Modul "queryModule"

Mit dem Tool "wsadmin" auf MBeans zugreifen

Sie können das in WebSphere Application Server bereitgestellte Dienstprogramm "wsadmin" verwenden, um auf MBean-Informationen zuzugreifen.

Führen Sie das Tool "wsadmin" im Verzeichnis "bin" Ihrer Installation von WebSphere Application Server aus. Im folgenden Beispiel wird eine Sicht der aktuellen Shard-Verteilung in einer dynamischen eXtreme-Scale-Umgebung abgerufen. Sie

können wsadmin in jeder Installation ausführen, in der eXtreme Scale ausgeführt wird. Das Dienstprogramm "wsadmin" muss nicht im Katalogservice ausgeführt werden.

```
$ wsadmin.sh -lang jython
wsadmin>placementService = AdminControl.queryNames
 ("com.ibm.websphere.objectgrid:*,type=PlacementService")
wsadmin>print AdminControl.invoke(placementService,
 "listObjectGridPlacement","library ms1")

<objectGrid name="library" mapSetName="ms1">
  <container name="container-0" zoneName="DefaultDomain"
  hostName="host1.company.org" serverName="server1">
     <shard type="Primary" partitionName="0"/>
     <shard type="SynchronousReplica" partitionName="1"/>
  </container>
  <container name="container-1" zoneName="DefaultDomain"
  hostName="host2.company.org" serverName="server2">
     <shard type="SynchronousReplica" partitionName="0"/>
     <shard type="Primary" partitionName="1"/>
  </container>
  <container name="UNASSIGNED" zoneName="_ibm_SYSTEM"
  hostName="UNASSIGNED" serverName="UNNAMED">
    <shard type="SynchronousReplica" partitionName="0"/>
    <shard type="AsynchronousReplica" partitionName="0"/>
  </container>
</objectGrid>
```

Überwachung mit Managed Beans (MBeans)

Sie können Managed Beans (MBeans) verwenden, um Statistiken in Ihrer Umgebung zu verfolgen.

Vorbereitende Schritte

Für die aufzuzeichnenden Attribute müssen Sie die Statistiken aktivieren. Sie können Statistiken mit den folgenden Methoden aktivieren:

- **Über die Servereigenschaftendatei:**

  Sie können Statistiken in der Servereigenschaftendatei mit einem Schlüssel/Wert-Eintrag im Format `statsSpec=<StatsSpec>` aktivieren. Es folgenden verschiedene Beispiele für mögliche Einstellungen:

  – Zum Aktivieren aller Statistiken verwenden Sie `statsSpec=all=enabled`.
  – Wenn Sie nur ObjectGrid-Statistiken aktivieren möchten, verwenden Sie `statsSpec=og.all=enabled`. Eine Beschreibung aller möglichen Statistikspezifikationen finden Sie in der API-Dokumentation zur APi "StatsSpec".

  Weitere Informationen zur Servereigenschaftendatei finden Sie im Abschnitt „Servereigenschaftendatei” auf Seite 188.

- **Mit einer Managed Bean:**

  Statistiken können jetzt mit dem Attribut " StatsSpec" der ObjectGrid-MBean aktiviert werden. Weitere Einzelheiten finden Sie in der Beschreibung der API StatsSpec.

- **Programmgesteuert:**

  Sie können Statistiken auch programmgesteuert mit der Schnittstelle "StatsAccessor" aktivieren, die über die Klasse " StatsAccessorFactory" abgerufen wird. Verwenden Sie diese Schnittstelle in einer Clientumgebung, oder wenn Sie eine Instanz von eXtreme Scale überwachen müssen, die im aktuellen Prozess ausgeführt wird.

Beispiel

Ein Beispiel für die Verwendung von Managed Beans finden Sie im Abschnitt „Überwachung mit dem Musterdienstprogramm "xsAdmin"" auf Seite 391.

Überwachung mit der Webkonsole

Eines der neuen Features im Release eXtreme Scale 7.1 ist eine Webkonsole, über die Sie aktuelle und Langzeitstatistiken grafisch darstellen können. Diese Konsole enthält einige vordefinierte Diagramme für allgemeine Übersichten und eine angepasste Berichtseite, die Sie verwenden können, um aus den verfügbaren Statistiken Diagramme zu erstellen. Sie können die Diagrammfunktionen in der Überwachungskonsole von WebSphere eXtreme Scale verwenden, um die allgemeine Leistung der Daten-Grids in Ihrer Umgebung anzuzeigen.

Vorbereitende Schritte

Der Konsolserver kann auf AIX-, Linux- oder Windows-Systemen ausgeführt werden. Das System mit dem Konsolserver muss in der Lage sein, eine Verbindung zum Katalogservice herzustellen, und der Katalogservice muss in der Lage sein, eine Verbindung zum Konsolserver herzustellen. Sie benötigen eine eigenständige Installation von WebSphere eXtreme Scale auf dem System mit dem Konsolserver. Das Script "startConsoleServer.bat|sh" für den Start des Konsolservers befindet sich im Verzeichnis "XS_ROOT/ObjectGrid/bin" Ihrer Installation.

Nachdem Sie Ihre Daten-Grids erstellt und Ihre Anwendungen für die Verwendung der Daten-Grids konfiguriert haben, müssen Sie eine Weile warten, bis die Statistiken verfügbar sind. Wenn Sie beispielsweise ein Daten-Grid für einen dynamischen Cache haben, sind die Statistiken erst verfügbar, wenn ein WebSphere Application Server mit einem dynamischen Cache eine Verbindung zum Daten-Grid für einen dynamischen Cache herstellt. Wenn Sie einen Verbund verwenden, muss die Initialisierung des Verbunds abgeschlossen sein, bevor Statistiken verfügbar sind. Im Allgemeinen müssen Sie nach einer größeren Konfigurationsänderung bis zu einer Minute warten, bis Sie die Änderungen in Ihren Statistiken sehen.

Tipp: Wenn Sie speziellere Informationen zu einem Datenpunkt im Diagramm anzeigen möchten, können Sie den Mauszeiger über diesen Datenpunkt bewegen.

Vorgehensweise

- Starten Sie den Konsolserver. Das Script "startConsoleServer.bat|sh" für den Start des Konsolservers befindet sich im Verzeichnis "XS_ROOT/ObjectGrid/bin" Ihrer Installation.
- Melden Sie sich an der Konsole an.
  1. Navigieren Sie im Webbrowser zu `https://Ihr.Konsol.Host:7443`. Ersetzen Sie `Ihr.Konsol.Host` durch den Hostnamen der Maschine, auf der Sie die Konsole installiert haben.
  2. Melden Sie sich an der Konsole an.
     - **Benutzer-ID:** `admin`
     - **Kennwort:** `admin`

     Die Begrüßungsseite der Konsole wird angezeigt.
  3. Klicken Sie auf **Einstellungen > Konfiguration**, um die Konsolkonfiguration zu prüfen. Die Konsolkonfiguration enthält Informationen wie die folgenden:
     - Trace-Zeichenfolge für den eXtreme-Scale-Client, z. B. `*=all=disabled`

- Administratorname und -kennwort
- E-Mail-Adresse des Administrators

- Zeigen Sie den Verbindungsstatus an.
  1. Navigieren Sie zur Begrüßungsseite, indem Sie in der Symbolleiste auf den Link **Home** klicken.
  2. Suchen Sie auf der rechten Seite der Symbolleiste die Dropdown-Liste, die die Domäne enthält, mit der der Konsolserver verbunden ist. Rechts von der Dropdown-Liste sehen Sie einen Verbindungsstatusanzeiger.
- Stellen Sie Verbindungen zu Katalogservern her, die Sie überwachen möchten, und verwalten Sie diese.
  1. Navigieren Sie zur Seite **Einstellungen > eXtreme-Scale-Katalogserver**.
  2. Fügen Sie einen neuen Katalogserver hinzu.
     a. Klicken Sie auf das Pluszeichen, um einen Dialog für die Registrierung eines vorhandenen Katalogservers anzuzeigen.
     b. Geben Sie Informationen wie den Hostnamen, den JMX-Port und den Listener-Port an.

        Hostname

        Zeigt den Hostnamen der Workstation an, auf der der Katalogservice ausgeführt wird.

        JMX-Port

        Zeigt die Nummer des Ports an, über den JMX/RMI-Verbindungen aktiviert werden. JMX ermöglicht die Überwachung und die Verwaltung über ferne Systeme.

        Listener-Port

        Zeigt den Listener-Port für die Kommunikation mit Internet Inter-ORB Protocol (IIOP).

     c. Klicken Sie auf **OK**.
     d. Vergewissern Sie sich, dass der Katalogserver der Navigationsstruktur hinzugefügt wurde.

  Zum Anzeigen von Informationen zu einem vorhandenen Katalogserver klicken Sie in der Navigationsstruktur auf der Seite **Einstellungen > eXtreme-Scale-Katalogserver** auf den Namen des Katalogservers.
- Stellen Sie Verbindungen zu Domänen her, die Sie überwachen möchten, und verwalten Sie diese.
  1. Navigieren Sie zur Seite **Einstellungen > eXtreme-Scale-Domänen**.
  2. Fügen Sie eine neue Domäne hinzu.
     a. Klicken Sie auf das Pluszeichen, um einen Dialog für die Registrierung einer Katalogservicedomäne anzuzeigen.
     b. Geben Sie Informationen an.

        Name Zeigt den vom Administrator zugeordneten Hostnamen der Domäne an.

        JMX-Benutzer
        Zeigt den verwendeten JMX-Benutzernamen (Java Management Extensions) an (falls die Sicherheit aktiviert ist).

        JMX-Kennwort
        Zeigt das JMX-Kennwort an, wenn die Sicherheit aktiviert ist.

Katalogserver
Listet die Katalogserver her, die zur ausgewählten Domäne gehören.

Generatorklasse
Zeigt den Namen der Klasse an, die die Schnittstelle "CredentialGenerator" implementiert. Diese Klasse wird verwendet, um Berechtigungsnachweise für Clients abzurufen.

Allgemeine Eigenschaften
Zeigt die Eigenschaften für die Implementierungsklasse "CredentialGenerator" an. Die Eigenschaften werden mit der Methode "setProperties(String)" für das Objekt gesetzt. Der Wert "credentialGeneratorprops" wird nur verwendet, wenn der Wert der Eigenschaft "credentialGeneratorClass" ungleich null ist.

c. Klicken Sie auf OK.
d. Vergewissern Sie sich, dass die Domäne der Navigationsstruktur hinzugefügt wurde.

3. Geben Sie auf dieser Seite die erforderlichen Sicherheitsinformationen an.
4. Ordnen Sie die Domäne Katalogservern zu, die Sie zuvor hinzugefügt haben.

Zum Anzeigen von Informationen zu einer vorhandenen Domäne klicken Sie in der Navigationsstruktur auf der Seite **Einstellungen > eXtreme-Scale-Domänen** auf den Namen der Domäne.

- Zum Anzeigen der aktuellen Serverstatistiken klicken Sie auf **Überwachen > Serverübersicht**.

  Serverstatistiken

  Genutzter Speicher
  Zeigt die derzeit (wirklich) genutzte Speicherkapazität in der Laufzeitumgebung des Servers an.

  Gesamtspeicher über der Zeit
  Zeigt die wirkliche Speichernutzung in der Laufzeitumgebung des Servers an.

  Genutzter Speicher über der Zeit
  Zeigt die Speichernutzung in der Laufzeitumgebung des Servers an.

- Zum Anzeigen der Leistung aller-Daten-Grids klicken Sie auf **Überwachen > Übersicht über die Daten-Grid-Domänen**.

  Übersichsstatistiken für Daten-Grid-Domänen

  Aktuelle Verteilung der genutzten Kapazität in Daten-Grids
  Dieses Diagramm enthält ein Bild des Gesamtpools und die Konsumenten mit der höchsten Nutzung an. Es werden nur die Top 25 Daten-Grids angezeigt.

  Top 5 Daten-Grids nach durchschnittlicher Transaktionsdauer in Millisekunden
  Dieses Diagramm enthält eine Liste der Top fünf Datencaches, organisiert nach durchschnittlicher Transaktionsdauer.

  Top 5 Daten-Grids nach durchschnittlichem Durchsatz in Transaktionen/Sekunde
  Dieses Diagramm enthält eine Liste der Top fünf Daten-Grids, organisiert nach durchschnittlichem Durchsatz, gemessen in Transaktionen pro Sekunde.

- Zum Anzeigen einzelner Daten-Grids klicken Sie auf **Überwachen > Übersicht über Daten-Grids > *Name_des_Daten-Grids*.** Auf dieser Seite wird eine Zusam-

menfassung angezeigt, die die Anzahl der Cacheeinträge, die durchschnittliche Transaktionsdauer und den durchschnittlichen Durchsatz enthält. Sie können die folgenden Diagramme anzeigen:

Übersichsstatistiken für Daten-Grids

Aktuelle Zusammenfassung
Zeigt Statistiken wie die aktuelle Anzahl der in diesem Grid zwischengespeicherten Objekte (Cacheeinträge), die durchschnittliche Transaktionsdauer und den durchschnittlichen Transaktionsdurchsatz an.

Genutzte Kapazität vs. Anzahl der Cacheeinträge
Dieses Diagramm zeigt die genutzte Kapazität des Caches im Vergleich mit der Anzahl der Einträge im Cache an. Sie können den angezeigten Zeitraum ändern: letzte Stunde, letzter Tag, letzte Woche, letzter Monat. Der Detaillierungsgrad der Anzeige variiert je nach ausgewähltem Zeitraum.

Gesamtanzahl der Cachabrufanforderungen vs. Erfolgreiche Cacheabrufanforderungen
Dieses Diagramm hilft Ihnen, die Anzahl erfolgreicher Cacheabfragen zu visualisieren.

- Wenn Sie weitere Details zu einem bestimmten Daten-Grid anzeigen möchten, klicken Sie auf **Überwachen > Details des Daten-Grids**. Es wird eine Baumstruktur mit allen Daten-Grids in Ihrer Konfiguration angezeigt. Sie können Die Detailabfrage/-analyse auf ein bestimmtes Daten-Grid beschränken, um nur die Maps dieses Daten-Grids anzuzeigen. Sie können auf den Namen eines Daten-Grids oder auf eine Map klicken, um weitere Informationen anzuzeigen:

  Daten-Grid-Statistiken

  Aktuelle Zusammenfassung
  Zeigt die aktuell genutzte Kapazität und eine Liste mit Zonen an, zu denen das Daten-Grid gehört.

  Aktuelle Verteilung der genutzten Kapazität in den eXtreme-Scale-ObjectGrids-Maps
  Zeigt einen Gesamtpool an, der die Kapazität nach Zone und die Gesamtkapazität in jeder Zone zeigt. Es werden nur die Top 25 ObjectGrid-Maps angezeigt.

  Aktuelle Verteilung der genutzten Kapazität in den Zonen
  Zeigt eine Zone an, die den Gesamtpool und die Hauptkonsumenten der Kapazität enthält. Es werden nur die Top 25 Zonen angezeigt.

  Map-Statistiken

  Aktuelle Zusammenfassung
  Zeigt Statistiken wie die folgenden an:

  Genutzte Kapazität
  Zeigt die genutzte Kapazität und eine Liste mit Zonen an, zu denen die Map gehört.

  Anzahl der Cacheeinträge
  Zeigt die Anzahl der in der Map zwischengespeicherten Objekte an.

  Durchschnittliche Transaktionsdauer (ms)
  Zeigt die durchschnittliche Fertigstellungszeit für Transaktionen an, an denen diese Map beteiligt ist.

Durchschnittlicher Transaktionsdurchsatz (Trans/Sek)
Zeigt die durchschnittliche Anzahl an Transaktionen pro Sekunde an, an denen diese Map beteiligt ist.

Aktuelle Verteilung der genutzten Kapazität in den Partitionen
Dieses Diagramm enthält ein Bild des Gesamtpools und die Konsumenten mit der höchsten Nutzung an. Es werden nur die Top 25 Partitionen angezeigt.

- Zur Auswahl der Statistiken für Ihren angepassten Bericht klicken Sie auf **Überwachen > Angepasste Berichte**.

  Verwenden Sie diese Sicht, um detaillierte Datendiagramme der verschiedenen Statistiken zu erstellen. Verwenden Sie die Baumstruktur, um die verfügbaren Daten-Grids und Server sowie die zugehörigen Statistiken zu untersuchen. Wenn Sie auf einen Knoten klicken bzw. die Eingabetaste drücken, wenn sich der Cursor auf einem Knoten befindet, der grafisch darstellbare Daten referenziert, erscheint ein Menü. Erstellen Sie ein neues Diagramm, das die Statistiken enthält, oder fügen Sie die Statistiken einem vorhandenen Diagramm mit kompatiblen Statistiken hinzu.

  Domänenstatistiken

  Durchschnittliche Transaktionsdauer (ms)
  Zeigt die durchschnittlich erforderliche Zeit für die Fertigstellung einer Transaktion in dieser Domäne an.

  Durchschnittlicher Transaktionsdurchsatz (Trans/Sek)
  Zeigt die durchschnittliche Anzahl an Transaktionen pro Sekunde in dieser Domäne an.

  Maximale Transaktionsdauer (ms)
  Zeigt die Dauer der längsten Transaktion in dieser Domäne an.

  Minimale Transaktionsdauer (ms)
  Zeigt die Dauer der kürzesten Transaktion in dieser Domäne an.

  Gesamttransaktionsdauer (ms)
  Zeigt die Gesamtdauer der Transaktionen in dieser Domäne seit der Initialisierung der Domäne an.

  eXtreme-Scale-Containerstatistiken

  Durchschnittliche Transaktionsdauer (ms)
  Zeigt die durchschnittlich erforderliche Zeit für die Fertigstellung einer Transaktion für diesen Katalogserver an.

  Durchschnittlicher Transaktionsdurchsatz (Trans/Sek)
  Zeigt die durchschnittliche Anzahl an Transaktionen pro Sekunde für diesen Katalogserver an.

  Maximale Transaktionsdauer (ms)
  Zeigt die Dauer der längsten Transaktion für diesen Katalogserver an.

  Minimale Transaktionsdauer (ms)
  Zeigt die Dauer der kürzesten Transaktion für diesen Katalogserver an.

  Gesamttransaktionsdauer (ms)
  Zeigt die Gesamtzeit an, die für Transaktionen für diesen Katalogserver seit dessen Initialisierung aufgebracht wurde.

  Gesamtanzahl der Einträge im Cache
  Zeigt die aktuelle Anzahl der in den Grids zwischengespeicherten Objekte an, die von diesem Katalogserver überwacht werden.

Maximale Anzahl der Einträge im Cache
Zeigt die maximale Anzahl der in den Grids zwischengespeicherten Objekte an, die von diesem Katalogserver überwacht werden.

Minimale Anzahl der Einträge im Cache
Zeigt die Mindestanzahl der in den Grids zwischengespeicherten Objekte an, die von diesem Katalogserver überwacht werden.

Trefferrate (Prozent)
Zeigt die Trefferrate (Trefferquote) für das ausgewählte Daten-Grid an. Eine hohe Trefferrate ist wünschenswert. Die Trefferrate zeigt an, wie dienlich das Grid bei der Vermeidung von Zugriffen auf den persistenten Speicher ist.

Belegung in Bytes
Zeigt die Speicherbelegung durch diese Map an.

Minimale Belegung in Bytes
Zeigt die niedrigste Speicherbelegung durch diesen Katalogservice und die zugehörigen Maps an.

Maximale Belegung in Bytes
Zeigt die höchste Speicherbelegung durch diesen Katalogservice und die zugehörigen Maps an.

Gesamtanzahl der Treffer
Zeigt an, wie oft die angeforderten Daten in der Map gefunden wurden und somit kein Zugriff auf den persistenten Speicher stattfinden musste.

Gesamtanzahl der Get-Anforderungen
Zeigt an, wie oft die Map auf den persistenten Speicher zugreifen musste, um Daten abzurufen.

Freier Heap-Speicher (MB)
Zeigt die tatsächliche Heap-Speicherkapazität an, die der vom Katalogserver verwendeten JVM zur Verfügung steht.

Gesamt-Heap-Speicher
Zeigt die Heap-Speicherkapazität an, die der von diesem Katalogserver verwendeten JVM zur Verfügung steht.

Genutzter Speicher
Zeigt die genutzte Speicherkapazität in der von diesem Katalogserver verwendeten JVM an.

Anzahl verfügbarer Prozessoren
Zeigt die Anzahl der CPUs, die diesem Katalogservice und dessen Maps zur Verfügung steht. Die höchste Stabilität erzielen Sie, wenn Sie Ihre Server mit einer Prozessorauslastung von 60 % und Ihre JVMs mit einer Heap-Speicherauslastung von 60 % betreiben. So können Lastspitzen die Prozessorauslastung auf 80–90 % hochtreiben. Ein dauerhafter Betrieb der Server mit diesen Ständen oder höher sollte aber vermieden werden.

Maximale Größe des Heap-Speichers (MB)
Zeigt die maximale Heap-Speicherkapazität an, die der von diesem Katalogserver verwendeten JVM zur Verfügung steht.

Grid-Statistiken

Durchschnittliche Transaktionsdauer (ms)
zeigt die durchschnittlich erforderliche Zeit für die Fertigstellung von Transaktionen an, an denen dieses Grid beteiligt ist.

Durchschnittlicher Transaktionsdurchsatz (Trans/Sek)
Zeigt die durchschnittliche Anzahl der von diesem Grid pro Sekunde ausgeführten Transaktionen an.

Maximale Transaktionsdauer (ms)
Zeigt die Dauer der längsten von diesem Grid ausgeführten Transaktion an.

Minimale Transaktionsdauer (ms)
Zeigt die Dauer der kürzesten von diesem Grid ausgeführten Transaktion an.

Gesamttransaktionsdauer (ms)
Zeigt die Gesamttransaktionsverarbeitungszeit für dieses Grid an.

Map-Statistiken

Gesamtanzahl der Einträge im Cache
Zeigt die aktuelle Anzahl der in dieser Map zwischengespeicherten Objekte an.

Maximale Anzahl der Einträge im Cache
Zeigt die maximale Anzahl zwischengespeicherter Objekt in dieser Map seit der Initialisierung der Map an.

Minimale Anzahl der Einträge im Cache
Zeigt die minimale Anzahl zwischengespeicherter Objekt in dieser Map seit der Initialisierung der Map an.

Trefferrate (Prozent)
Zeigt die Trefferrate (Trefferquote) für die ausgewählte Map an. Eine hohe Trefferrate ist wünschenswert. Die Trefferrate zeigt an, wie dienlich die Map bei der Vermeidung von Zugriffen auf den persistenten Speicher ist.

Belegung in Bytes
Zeigt die Speicherbelegung durch diese Map an.

Minimale Belegung in Bytes
Zeigt die minimale Belegung (in Bytes) für diese Map an.

Maximale Belegung in Bytes
Zeigt die maximale Belegung (in Bytes) für diese Map an.

Gesamtanzahl der Treffer
Zeigt an, wie oft die angeforderten Daten in der Map gefunden wurden und somit kein Zugriff auf den persistenten Speicher stattfinden musste.

Gesamtanzahl der Get-Anforderungen
Zeigt an, wie oft die Map auf den persistenten Speicher zugreifen musste, um Daten abzurufen.

Freier Heap-Speicher (MB)
Zeigt die aktuell verfügbare Heap-Speicherkapazität für diese Map in der vom Katalogserver verwendeten JVM an.

Gesamt-Heap-Speicher (MB)
Zeigt die insgesamt verfügbare Heap-Speicherkapazität für diese Map in der vom Katalogserver verwendeten JVM an. Die höchste Stabilität erzielen Sie, wenn Sie Ihre Server mit einer Prozessorauslastung von 60 % und Ihre JVMs mit einer Heap-Speicherauslastung von 60 % betreiben. So können Lastspitzen die Prozessorauslastung auf 80–90 % hochtreiben. Ein dauerhafter Betrieb der Server mit diesen Ständen oder höher sollte aber vermieden werden.

Genutzter Speicher (MB)
Zeigt den genutzten Speicher in dieser Map an.

Anzahl verfügbarer Prozessoren
Zeigt die Anzahl der CPUs an, die dieser Map zur Verfügung steht. Die höchste Stabilität erzielen Sie, wenn Sie Ihre Server mit einer Prozessorauslastung von 60 % und Ihre JVMs mit einer Heap-Speicherauslastung von 60 % betreiben. So können Lastspitzen die Prozessorauslastung auf 80–90 % hochtreiben. Ein dauerhafter Betrieb der Server mit diesen Ständen oder höher sollte aber vermieden werden.

Maximale Größe des Heap-Speichers (MB)
Zeigt die maximal verfügbare Heap-Speicherkapazität für diese Map in der vom Katalogserver verwendeten JVM an.

Überwachung mit Tools eines anderen Anbieters

WebSphere eXtreme Scale kann mit verschiedenen gängigen Lösungen für die Unternehmensüberwachung überwacht werden. Es werden Plug-in-Agenten für IBM Tivoli Monitoring and Hyperic HQ bereitgestellt, die WebSphere eXtreme Scale mit Hilfe öffentlich zugänglicher Management-Beans überwachen. CA Wily Introscope verwendet die Java-Methodeninstrumentierung, um Statistiken zu erfassen.

Überwachung mit IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale

IBM Tivoli Enterprise Monitoring Agent ist eine mit vielen Funktionen ausgestattete Überwachungslösung, die Sie verwenden können, um Datenbanken, Betriebssysteme und Server in verteilten Umgebungen und in Hostumgebungen zu überwachen. WebSphere eXtreme Scale enthält einen angepassten Agenten, den Sie verwenden können, um eXtreme-Scale-Management-Beans selbst zu überwachen. Diese Lösung funktioniert effizient in eigenständigen Implementierungen von eXtreme Scale und in Implementierungen von eXtreme Scale mit WebSphere Application Server.

Vorbereitende Schritte

- Installieren Sie WebSphere eXtreme Scale Version 7.0.0 oder höher.

  Außerdem müssen Statistiken aktiviert werden, um statistische Daten von eXtreme-Scale-Servern zu erfassen. Verschiedene Optionen zum Aktivieren von Statistiken sind in „Überwachung mit Managed Beans (MBeans)” auf Seite 406 und „Überwachung mit dem Musterdienstprogramm "xsAdmin"” auf Seite 391 beschrieben.
- Installieren Sie IBM Tivoli Monitoring Version 6.2.1 mit Fixpack 2 oder höher.
- Installieren Sie den Tivoli-Betriebssystemagenten auf jedem Server oder Host, auf dem eXtreme-Scale-Server ausgeführt werden.
- Installieren Sie den WebSphere eXtreme Scale-Agenten, den Sie kostenlos von der Website von IBM Open Process Automation Library (OPAL) herunterladen können.

Führen Sie die folgenden Schritte aus, um Tivoli Monitoring Agent zu installieren und zu konfigurieren:

Vorgehensweise

1. Installieren Sie Tivoli Monitoring Agent for WebSphere eXtreme Scale.

Laden Sie das Tivoli-Installations-Image herunter, und entpacken Sie die Dateien in einem temporären Verzeichnis.

2. Installieren Sie die Unterstützungsdateien für eXtreme-Scale-Anwendungen.

   Installieren Sie die eXtreme-Scale-Anwendungsunterstützung in jeder der folgenden Implementierungen:

   - Tivoli Enterprise Portal Server (TEPS)
   - Enterprise Desktop Client (TEPD)
   - Tivoli Enterprise Monitoring Server (TEMS)

   a. Starten Sie in dem erstellten temporären Verzeichnis ein neues Befehlsfenster, und führen Sie die entsprechende ausführbare Datei für Ihre Plattform aus. Das Installationsscript erkennt den Typ Ihrer Tivoli-Implementierung (TEMS, TEPD oder TEPS) automatisch. Sie können jeden beliebigen Typ auf einem einzelnen oder auf mehreren Hosts installieren, und alle drei Implementierungstypen erfordern die Installation der Unterstützungsdateien für die eXtreme-Scale-Agentenanwendung.

   b. Vergewissern Sie sich, dass die ausgewählten Optionen für die implementierten Tivoli-Komponenten im Fenster des Installationsprogramms korrekt sind. Klicken Sie auf **Next**.

   c. Geben Sie auf Anforderung Ihren Hostnamen und Ihre Verwaltungsberechtigungsnachweise an. Klicken Sie auf **Next**.

   d. Wählen Sie **Monitoring Agent for WebSphere eXtreme Scale** aus. Klicken Sie auf **Next**.

   e. Sie werden über die auszuführenden Installationsaktionen benachrichtigt. Klicken Sie auf **Next**. Der Fortschritt der Installation wird bis zum Ende hin angezeigt.

   Nach der Ausführung dieser Prozedur sind alle erforderlichen Anwendungsunterstützungsdateien für den eXtreme-Scale-Agenten installiert.

3. Installieren Sie den Agenten auf jedem eXtreme-Scale-Knoten.

   Sie installieren einen Tivoli-Betriebssystemagenten auf jedem Computer. Sie müssen diesen Agenten nicht konfigurieren oder starten. Verwenden Sie dasselbe Installations-Image, das Sie bereits im vorherigen Schritt verwendet haben, um die plattformspezifische ausführbare Datei auszuführen.

   Als Richtlinie können Sie verwenden, dass nur ein einziger Agent pro Host installiert werden muss. Jeder Agent kann mehrere Instanzen von eXtreme-Scale-Servern unterstützen. Um die beste Leistung zu erzielen, verwenden Sie eine Agenteninstanz für ungefähr 50 eXtreme-Scale-Server.

   a. Klicken Sie in der Eingangsanzeige des Installationsassistenten auf **Next**, um die Anzeige zu öffnen, in der Sie Informationen zum Installationspfad angeben.

   b. Geben Sie im Feld **Tivoli Monitoring installation directory** einen Wert ein, oder navigieren Sie zum Verzeichnis `C:\IBM\ITM` (oder `/opt/IBM/ITM`). Vergewissern Sie sich, dass der im Feld **Location for installable media** angezeigte Wert korrekt ist, und klicken Sie auf **Next**.

   c. Wählen Sie die Komponenten aus, die Sie hinzufügen möchten, z. B. **Perform a local install of the solution**, und klicken Sie auf **Next**.

   d. Wählen Sie die Anwendungen aus, für die Sie die Unterstützung hinzufügen möchten, z. B. **Monitoring Agent for WebSphere eXtreme Scale**, und klicken Sie auf **Next**.

   e. Der Fortschritt der Installation wird angezeigt, bis alle Anwendungsunterstützungsdateien erfolgreich hinzugefügt wurden.

Anmerkung: Wiederholen Sie diese Schritte auf jedem eXtreme-Scale-Knoten. Sie können auch eine unbeaufsichtigte Installation durchführen. Weitere Informationen zur unbeaufsichtigten Installation finden Sie im Information Center von IBM Tivoli Monitoring.

4. Konfigurieren Sie den WebSphere eXtreme Scale-Agenten.

   Jeder installierte Agent muss für die Überwachung eines Katalogservers und oder eXtreme-Scale-Servers konfiguriert werden.

   Die Schritte zum Konfigurieren von Windows- und UNIX-Plattformen sind verschieden. Die Konfiguration für die Windows-Plattform erfolgt über die Benutzerschnittstelle **Manage Tivoli Monitoring Services**. Die Konfiguration für UNIX-Plattformen ist befehlszeilenbasiert.

   Windows Verwenden Sie die folgenden Schritte, um den Agenten anfänglich unter Windows zu konfigurieren.

   a. Klicken Sie im Fenster **Manage Tivoli Enterprise Monitoring Services** auf **Start → All Programs → IBM Tivoli Monitoring → Manage Tivoli Monitoring Services**.
   b. Klicken Sie mit der rechten Maustaste auf **Monitoring Agent for WebSphere eXtreme Scale**, und wählen Sie die Option **Configure using default** aus, woraufhin ein Fenster erscheint, in dem Sie eine eindeutige Instanz des Agenten erstellen können.
   c. Wählen Sie einen eindeutigen Namen aus, z. B. `instance1`, und klicken Sie auf **Next**.

   • Wenn Sie eigenständige eXtreme-Scale-Server überwachen möchten, führen Sie die folgenden Schritte aus:

   a. Aktualisieren Sie die Java-Parameter, und stellen Sie sicher, dass der Wert für **Java Home** korrekt ist. JVM-Argumente können leer bleiben. Klicken Sie auf **Next**.
   b. Wählen Sie für **MBean server connection type** einen Typ aus. Verwenden Sie `JSR-160-Complaint Server` für eigenständige eXtreme-Scale-Server. Klicken Sie auf **Next**.
   c. Wenn die Sicherheit aktiviert ist, aktualisieren Sie die Werte in den Feldern **User ID** und **Password**. Übernehmen Sie den Wert im Feld **JMX service URL**. Sie überschreiben diesen Wert später. Übernehmen Sie den Wert im Feld **JMX Class Path Information**. Klicken Sie auf **Next**.

   Zum Konfigurieren der Server für den Agenten unter Windows führen Sie die folgenden Schritte aus:

   a. Konfigurieren Sie Unterknoteninstanzen von eXtreme-Scale-Servern im Teilfenster **WebSphere eXtreme Scale Grid Servers**. Wenn keine Containerserver auf Ihrem Computer vorhanden sind, klicken Sie auf **Next**, um das Teilfenster für den Katalogservice aufzurufen.
   b. Wenn mehrere eXtreme-Scale-Containerserver auf Ihrem Computer vorhanden sind, konfigurieren Sie den Agenten so, dass jeder einzelne Server überwacht wird.
   c. Sie können so viele eXtreme-Scale-Server hinzufügen, wie Sie benötigen, sofern ihre Namen und Ports eindeutig sind. Klicken Sie dazu auf **New**. (Wenn ein eXtreme-Scale-Server gestartet wird, muss ein Wert für den JMX-Port angegeben werden.)
   d. Nach der Konfiguration der Containerserver klicken Sie auf **Next**. Daraufhin wird das Teilfenster **WebSphere eXtreme Scale Catalog Servers** angezeigt.

e. Wenn Sie keine Katalogserver haben, klicken Sie auf **OK**. Wenn Sie Katalogserver haben, fügen Sie eine neue Konfiguration für jeden Server hinzu, so wie Sie es für die Containerserver getan haben. Wählen Sie auch hier einen eindeutigen Namen aus, vorzugsweise denselben Namen, den Sie auch beim Starten des Katalogservers verwendet haben. Klicken Sie auf **OK**, um die Konfiguration zu beenden.

- Wenn Sie Server für den Agenten in eXtreme-Scale-Servern überwachen möchten, die in einen Prozess von WebSphere Application Server integriert sind, führen Sie die folgenden Schritte aus:
  a. Aktualisieren Sie die Java-Parameter, und stellen Sie sicher, dass der Wert für **Java Home** korrekt ist. JVM-Argumente können leer bleiben. Klicken Sie auf **Next**.
  b. Wählen Sie im Feld **MBean server connection type** einen Typ aus. Wählen Sie die Version von WebSphere Application Server für Ihre Umgebung aus. Klicken Sie auf **Next**.
  c. Stellen Sie sicher, dass die Informationen zu WebSphere Application Server in dieser Anzeige korrekt sind. Klicken Sie auf **Next**.
  d. Fügen Sie nur eine einzige Unterknotendefinition hinzu. Legen Sie einen Namen für die Unterknotendefinition fest, aber aktualisieren Sie die Portdefinition nicht. In einer Umgebung mit WebSphere Application Server können Daten von allen Anwendungsserverprozessen erfasst werden, die vom Node Agent verwaltet werden, der auf dem Computer ausgeführt wird. Klicken Sie auf **Next**.
  e. Wenn keine Katalogserver in der Umgebung vorhanden sind, klicken Sie auf **OK**. Wenn Sie Katalogserver haben, fügen Sie eine neue Konfiguration für jeden Katalogserver hinzu, ähnlich wie bei den Containerservern. Wählen Sie einen eindeutigen Namen für den Katalogservice aus, vorzugsweise den Namen, den Sie auch beim Starten des Katalogservice verwendet haben. Klicken Sie auf **OK**, um die Konfiguration zu beenden.

Anmerkung: Die Containerserver müssen nicht zusammen mit dem Katalogservice in einem Prozess ausgeführt werden.

Jetzt sind der Agent und die Server konfiguriert und betriebsbereit. Klicken Sie im nächsten Fenster mit der rechten Maustaste auf `instance1`, um den Agenten zu starten.

UNIX Zum Konfigurieren des Agenten auf der UNIX-Plattform über die Befehlszeile, führen Sie die folgenden Schritte aus:

Es folgt ein Beispiel für eigenständige Server, die einen JSR160-konformen Verbindungstyp verwenden. Das Beispiel zeigt drei eXtreme-Scale-Container auf dem einzelnen Host (rhea00b02), und die JMX-Listener-Adressen sind 15000,15001 und 15002. Es gibt keine Katalogserver.

Die Ausgabe des Konfigurationsdienstprogramms wird in *Kursivschrift mit fester Breite* angezeigt und die Benutzeraktion in **Fettschrift mit fester Breite**. (Wenn keine Benutzeraktion erforderlich ist, wird der Standardwert durch Drücken der Eingabetaste ausgewählt.)

```
rhea00b02 # ./itmcmd config -A xt
Agent configuration started...
Enter instance name (default is: ): inst1
Edit "Monitoring Agent for WebSphere eXtreme Scale" settings? [ 1=Yes, 2=No ] (default is: 1):
Edit 'Java' settings? [ 1=Yes, 2=No ] (default is: 1):
Java home (default is: C:\Program Files\IBM\Java50): /opt/OG61/java
Java trace level [ 1=Error, 2=Warning, 3=Information, 4=Minimum Debug, 5=Medium Debug, 6=Maximum Debug,
    7=All ] (default is: 1):
JVM arguments (default is: ):
Edit 'Connection' settings? [ 1=Yes, 2=No ] (default is: 1):
MBean server connection type [ 1=JSR-160-Compliant Server, 2=WebSphere Application Server version 6.0,
```

```
3=WebSphere Application Server version 6.1, 4=WebSphere Application Server version 7.0 ] (default is: 1): 1
Edit 'JSR-160-Compliant Server' settings? [ 1=Yes, 2=No ] (default is: 1):
JMX user ID (default is: ):
Enter JMX password (default is: ):
Re-type : JMX password (default is: ):
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:port/objectgrid/MBeanServer):
----------------------------------------
JMX Class Path Information
JMX base paths (default is: ):
JMX class path (default is: ):
JMX JAR directories (default is: ):
Edit 'WebSphere eXtreme Scale Catalog Service' settings? [ 1=Yes, 2=No ] (default is: 1): 2
Edit 'WebSphere eXtreme Scale Grid Servers' settings? [ 1=Yes, 2=No ] (default is: 1): 1
No 'WebSphere eXtreme Scale Grid Servers' settings available?
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 1
WebSphere eXtreme Scale Grid Servers (default is: ): rhea00b02_c0
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer):
service:jmx:rmi:///jndi/rmi://localhost:15000/objectgrid/MBeanServer

'WebSphere eXtreme Scale Grid Servers' settings:  WebSphere eXtreme Scale Grid Servers=ogx
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 1
WebSphere eXtreme Scale Grid Servers (default is: ): rhea00b02_c1
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer):
service:jmx:rmi:///jndi/rmi://localhost:15001/objectgrid/MBeanServer

'WebSphere eXtreme Scale Grid Servers' settings:  WebSphere eXtreme Scale Grid Servers= rhea00b02_c1
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 1
WebSphere eXtreme Scale Grid Servers (default is: ): rhea00b02_c2
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer):
service:jmx:rmi:///jndi/rmi://localhost:15002/objectgrid/MBeanServer

'WebSphere eXtreme Scale Grid Servers' settings:  WebSphere eXtreme Scale Grid Servers= rhea00b02_c2
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 5

Will this agent connect to a TEMS? [1=YES, 2=NO] (Default is: 1):
TEMS Host Name (Default is: rhea00b00):

Network Protocol [ip, sna, ip.pipe or ip.spipe] (Default is: ip.pipe):

     Now choose the next protocol number from one of these:
     - ip
     - sna
     - ip.spipe
     - 0 for none
Network Protocol 2 (Default is: 0):
IP.PIPE Port Number (Default is: 1918):
Enter name of KDC_PARTITION (Default is: null):

Configure connection for a secondary TEMS? [1=YES, 2=NO] (Default is: 2):
Enter Optional Primary Network Name or 0 for "none" (Default is: 0):
Agent configuration completed...
```

Der vorherige Beispielcode erstellt eine Agenteninstanz mit dem Namen "inst1" und aktualisiert die Einstellungen für das Java-Ausgangsverzeichnis. Die eXtreme-Scale-Containerserver sind konfiguriert, aber der Katalogserver ist nicht konfiguriert.

Anmerkung: Die vorherige Prozedur erstellt eine Textdatei im folgenden Format im Verzeichnis `<ITM-Installation>/config/<Host>_xt_<Instanzname>.cfg`.

Beispiel: `rhea00b02_xt_inst1.cfg`

Es empfiehlt sich, diese Datei mit einem Texteditor Ihrer Wahl zu editieren. Ein Beispiel für den Inhalt einer solchen Datei folgt:

```
INSTANCE=inst2 [        SECTION=KQZ_JAVA [ { JAVA_HOME=/opt/OG61/java }  { JAVA_TRACE_LEVEL=ERROR }  ]
SECTION=KQZ_JMX_CONNECTION_SECTION [ { KQZ_JMX_CONNECTION_PROPERTY=KQZ_JMX_JSR160_JSR160 }  ]
SECTION=KQZ_JMX_JSR160_JSR160 [ { KQZ_JMX_JSR160_JSR160_CLASS_PATH_TITLE= }
{ KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:rmi:///jndi/rmi://localho
st:port/objectgrid/MBeanServer }  { KQZ_JMX_JSR160_JSR160_CLASS_PATH_SEPARATOR= }  ]
SECTION=OGS:rhea00b02_c1 [ { KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:
rmi:///jndi/rmi://localhost:15001/objectgrid/MBeanServer }  ]
SECTION=OGS:rhea00b02_c0 [ { KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:
rmi:///jndi/rmi://localhost:15002/objectgrid/MBeanServer }  ]
SECTION=OGS:rhea00b02_c2 [ { KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:
rmi:///jndi/rmi://localhost:15002/objectgrid/MBeanServer }  ]]
```

Im Folgenden sehen Sie ein Beispiel, das die Konfiguration einer Implementierung von WebSphere Application Server zeigt:

```
rhea00b02 # ./itmcmd config -A xt
Agent configuration started...
Enter instance name (default is: ): inst1
Edit "Monitoring Agent for WebSphere eXtreme Scale" settings? [ 1=Yes, 2=No ] (default is: 1): 1
Edit 'Java' settings? [ 1=Yes, 2=No ] (default is: 1): 1
Java home (default is: C:\Program Files\IBM\Java50): /opt/WAS61/java
Java trace level [ 1=Error, 2=Warning, 3=Information, 4=Minimum Debug, 5=Medium Debug, 6=Maximum Debug,
    7=All ] (default is: 1):
JVM arguments (default is: ):
Edit 'Connection' settings? [ 1=Yes, 2=No ] (default is: 1):
MBean server connection type [ 1=JSR-160-Compliant Server, 2=WebSphere Application Server version 6.0,
3=WebSphere Application Server version 6.1, 4=WebSphere Application Server version 7.0 ] (default is: 1): 4
Edit 'WebSphere Application Server version 7.0' settings? [ 1=Yes, 2=No ] (default is: 1):WAS user ID (default is: ):
Enter WAS password (default is: ):
Re-type : WAS password (default is: ):
WAS host name (default is: localhost): rhea00b02
WAS port (default is: 2809):
WAS connector protocol [ 1=rmi, 2=soap ] (default is: 1):
WAS profile name (default is: ): default
----------------------------------------
WAS Class Path Information
WAS base paths (default is: C:\Program Files\IBM\WebSphere\AppServer;/opt/IBM/WebSphere/AppServer): /opt/WAS61
WAS class path (default is: runtimes/com.ibm.ws.admin.client_6.1.0.jar;runtimes/com.ibm.ws.ejb.thinclient_7.0.0.jar):
WAS JAR directories (default is: lib;plugins):
Edit 'WebSphere eXtreme Scale Grid Servers' settings? [ 1=Yes, 2=No ] (default is: 1):
No 'WebSphere eXtreme Scale Grid Servers' settings available?
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 1
WebSphere eXtreme Scale Grid Servers (default is: ): rhea00b02
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer):

'WebSphere eXtreme Scale Grid Servers' settings:  WebSphere eXtreme Scale Grid Servers=rhea00b02
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 5
Edit 'WebSphere eXtreme Scale Catalog Service' settings? [ 1=Yes, 2=No ] (default is: 1): 2
Will this agent connect to a TEMS? [1=YES, 2=NO] (Default is: 1):
TEMS Host Name (Default is: rhea00b02):

Network Protocol [ip, sna, ip.pipe or ip.spipe] (Default is: ip.pipe):

     Now choose the next protocol number from one of these:
     - ip
     - sna
     - ip.spipe
     - 0 for none
Network Protocol 2 (Default is: 0):
IP.PIPE Port Number (Default is: 1918):
Enter name of KDC_PARTITION (Default is: null):

Configure connection for a secondary TEMS? [1=YES, 2=NO] (Default is: 2):
Enter Optional Primary Network Name or 0 for "none" (Default is: 0):
Agent configuration completed...
rhea00b02 #
```

Für Implementierungen von WebSphere Application Server müssen Sie nicht mehrere Unterknoten erstellen. Der eXtreme-Scale-Agent stellt die Verbindung zum Node Agent her, um alle Informationen von den Anwendungsservern zu erfassen, für die er zuständig ist.

`SECTION=CAT` bezeichnet eine Katalogservicezeile, wohingegen `SECTION=OGS` eine Konfigurationszeile für einen eXtreme-Scale-Server bezeichnet.

5. Konfigurieren Sie den JMX-Port für alle Containerserver von eXtreme Scale.

   Wenn eXtreme-Scale-Containerserver ohne das Argument **-JMXServicePort** gestartet werden, wird einem MBean-Server ein dynamischer Port zugeordnet. Der Agent muss im Voraus wissen, mit welchem Port er zu kommunizieren hat. Der Agent funktioniert nicht mit dynamischen Ports.

   Wenn Sie die Server starten, müssen Sie das Argument `-JMXServicePort <Portnummer>` angeben, wenn Sie den eXtreme-Scale-Server mit dem Befehl

startOgServer.sh | .bat starten. Die Ausführung dieses Befehls gewährleistet, dass der JMX-Server im Prozess an einem statischen vordefinierten Port empfangsbereit ist.

Für die vorherigen Beispiele müssen in einer UNIX-Installation zwei eXtreme-Scale-Server mit definierten Ports gestartet werden:

a. "-JMXServicePort" "15000" (für rhea00b02_c0)
b. "-JMXServicePort" "15001" (für rhea00b02_c1)

a. Starten Sie den eXtreme Scale-Agenten.

   Davon ausgehend, dass wie im vorherigen Beispiel die Instanz inst1 erstellt wurde, setzen Sie die folgenden Befehle ab:

   1) `cd <ITM-Installation>/bin`
   2) `itmcmd agent –o inst1 start xt`

b. Stoppen Sie den eXtreme Scale-Agenten.

   Davon ausgehend, dass wie im vorherigen Beispiel die Instanz inst1 erstellt wurde, setzen Sie die folgenden Befehle ab:

   1) `cd <ITM-Installation>/bin`
   2) `itmcmd agent –o inst1 stop xt`

6. Aktivieren Sie Statistiken für alle Containerserver von eXtreme Scale.

   Der Agent verwendet die Statistik-MBeans von eXtreme Scale, um Statistiken aufzuzeichnen. Die Statistikspezifikation von eXtreme Scale muss mit einer der folgenden Methoden aktiviert werden.

   - Konfiguration von Servereigenschaften für die Aktivierung aller Statistiken, wenn die Containerserver gestartet werden: all=enabled.
   - Verwendung des Musterdienstprogramms xsadmin für die Aktivierung von Statistiken für alle aktiven Container mit den Parametern "-setstatsspec all=enabled"

Ergebnisse

Nachdem alle Server konfiguriert und gestartet wurden, werden MBean-Daten in der Konsole von IBM Tivoli Portal angezeigt. In vordefinierten Arbeitsbereichen werden Graphen und Datenmetriken auf jeder Knotenebene angezeigt.

Die folgenden Arbeitsbereiche sind definiert: **eXtreme Scale Grid Servers** - für alle überwachten Knoten

- eXtreme Scale Transactions View
- eXtreme Scale Primary Shard View
- eXtreme Scale Memory View
- eXtreme Scale ObjectMap View

Sie können auch eigene Arbeitsbereiche konfigurieren. Weitere Informationen finden Sie in den Informationen zum Anpassen von Arbeitsbereichen im Information Center von IBM Tivoli Monitoring.

eXtreme-Scale-Anwendungen mit CA Wily Introscope überwachen

CA Wily Introscope ist ein Managementprodukt eines anderen Anbieters, das Sie verwenden können, um Leistungsprobleme in Unternehmensanwendungsumgebungen zu erkennen und zu diagnostizieren. eXtreme Scale enthält Einstellungen für die Konfiguration von CA Wily Introscope für die Introspektion ausgewählter

Komponenten der Laufzeitumgebung von eXtreme Scale, um eXtreme-Scale-Anwendungen schnell anzeigen und validieren zu können. CA Wily Introscope funktioniert effizient für eigenständige Implementierungen und Implementierungen von WebSphere Application Server.

Übersicht

Wenn Sie eXtreme-Scale-Anwendungen mit CA Wily Introscope überwachen möchten, müssen Sie Einstellungen in den PBD-Dateien (ProbeBuilderDirective) festlegen, die Ihnen Zugriff auf die Überwachungsinformationen für eXtreme Scale geben.

Achtung: Die Instrumentierungspunkte für Introscope können sich mit jedem Fixpack oder Release ändern. Wenn Sie ein neues Fixpack oder Release installieren, suchen Sie in der Dokumentation nach Hinweisen zu Änderungen bezüglich der Instrumentierungspunkte.

Sie können PBD-Dateien (ProbeBuilderDirective) von CA Wily Introscope konfigurieren, um Ihre eXtreme-Scale-Anwendungen zu überwachen. CA Wily Introscope ist ein Anwendungsmanagementprodukt, mit dem Sie Leistungsprobleme in komplexen, Verbund- und Webanwendungsumgebungen proaktiv erkennen, sichten und diagnostizieren können.

Einstellungen in der PBD-Datei für die Überwachung des Katalogservice

Sie können eine oder mehrere der folgenden Einstellungen in Ihrer PBD-Datei verwenden, um den Katalogservice zu überwachen:

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  changeDefinedCompleted
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  viewChangeCompleted
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  viewAboutToChange
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeat
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatCluster
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatCurrentLeader
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatDeadServer
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatNewLeader
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatNewServer
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  PlacementServiceImpl
importRouteInfo BlamePointTracerDifferentMethods
  "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  PlacementServiceImpl heartbeat
```

```
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  PlacementServiceImpl joinPlacementGroup
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
  TraceOneMethodOfClass:
com.ibm.ws.objectgrid.catalog.placement.PlacementServiceImpl
  classifyServer
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  BalanceGridEventListener shardActivated
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  BalanceGridEventListener shardDeactivate
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
```

Klassen für die Überwachung des Katalogservice

HAControllerImpl
Die Klasse "HAControllerImpl" verarbeitet Lebenszyklus- und Feedback-Ereignisse für die Stammgruppe. Sie können diese Klasse überwachen, um einen Hinweis auf die Struktur und Änderungen der Stammgruppe zu erhalten.

ServerAgent
Die Klasse "ServerAgent" ist für die Kommunikation von Stammgruppenereignissen an den Katalogservice zuständig. Sie können die verschiedenen Aufrufe für den Austausch von Überwachungssignalen überwachen, um wichtige Ereignisse zu erkennen.

PlacementServiceImpl
Die Klasse "PlacementServiceImpl" koordiniert die Container. Sie können die Methoden in dieser Klasse verwenden, um das Beitreten von Servern zur Stammgruppe und Verteilungsereignisse zu überwachen.

BalanceGridEventListener
Die Klasse "BalanceGridEventListener" steuert die Leitung des Katalogs. Sie können diese Klasse überwachen, um einen Hinweis auf den Katalogservice zu erhalten, der momentan als leitender Server agiert.

Einstellungen in der PBD-Datei für die Überwachung der Container

Sie können eine oder mehrere der folgenden Einstellungen in Ihrer PBD-Datei verwenden, um die Container zu überwachen:

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.ShardImpl processMessage
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.plugins.
  CommittedLogSequenceListenerProxy applyCommitted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.plugins.
  CommittedLogSequenceListenerProxy sendApplyCommitted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.map.BaseMap evictMapEntries
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.checkpoint.
  CheckpointMapImpl$CheckpointIterator activateListener
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  changeDefinedCompleted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  viewChangeCompleted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
```

```
  viewAboutToChange
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  batchProcess
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeat
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatCluster
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatCurrentLeader
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatDeadServer
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatNewLeader
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatNewServer
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
```

Klassen für die Überwachung der Container

ShardImpl
Die Klasse "ShardImpl" hat die Methode "processMessage. Die Methode "processMessage" ist die Methode für Clientanforderungen. Mit dieser Methode können Sie serverseitige Antwortzeiten und Anforderungszähler abrufen. Indem Sie die Zähler für alle Server und die Auslastung des Heap-Speichers überwachen, können Sie feststellen, ob das Grid ausgeglichen ist.

CheckpointIterator
Die Klasse "CheckpointIterator" hat die Methode "activateListener", die primäre Shards in den Peer-Modus versetzt. Wenn die primären Shards in den Peer-Modus versetzt werden, ist das Replikat nach Abschluss der Methode auf demselben Stand wie das primäre Shard. Wenn ein Replikat über ein vollständiges primäres Shard neu generiert wird, kann diese Operation längere Zeit dauern. Das System ist erst dann vollständig wiederhergestellt, wenn diese Operation abgeschlossen ist. Sie können diese Klasse verwenden, um den Fortschritt der Operation zu überwachen.

CommittedLogSequenceListenerProxy
Die Klasse "CommittedLogSequenceListenerProxy" hat zwei Methoden, die von Interesse sind. Die Methode "applyCommitted" wird für jede Transaktion ausgeführt, und die Methode "sendApplyCommitted" wird ausgeführt, wenn das Replikat Informationen extrahiert. Das Verhältnis, in dem die beiden Methoden ausgeführt werden, kann Ihnen einen Hinweis darauf geben, inwieweit das Replikat in der Lage ist, mit dem primären Shard Schritt zu halten.

Einstellungen in der PBD-Datei für die Überwachung der Clients

Sie können eine oder mehrere der folgenden Einstellungen in Ihrer PBD-Datei verwenden, um die Clients zu überwachen:

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.client.ORBClientCoreMessageHandler
 sendMessage
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.corba.cluster.ClusterStore
  bootstrap
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.corba.cluster.ClusterStore
```

```
  epochChangeBootstrap
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.map.BaseMap evictMapEntries
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.cluster.orb.routing.
  SelectionServiceImpl routeFailed
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.cluster.orb.routing.
  SelectionServiceImpl routeFailed
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.SessionImpl getMap
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.ObjectGridImpl getSession
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TurnOn: ObjectMap
SetFlag: ObjectMap
IdentifyClassAs: com.ibm.ws.objectgrid.ObjectMapImpl ObjectMap
TraceComplexMethodsifFlagged: ObjectMap BlamePointTracerDifferentMethods
"OGclient|{classname}|{method}"
```

Klassen für die Überwachung der Clients

ORBClientCoreMessageHandler
: Die Klasse "ORBClientCoreMessageHandler" ist für das Senden von Anwendungsanforderungen an die Container zuständig. Sie können die Methode "sendMessage" überwachen, um Clientantwortzeiten und Anforderungsanzahl zu erhalten.

ClusterStore
: Die Klasse "ClusterStore" enthält Routing-Informationen auf der Clientseite.

BaseMap
: Die Klasse "BaseMap" hat die Methode "evictMapEntries", die aufgerufen wird, wenn der Evictor Einträge aus der Map entfernen möchte.

SelectionServiceImpl
: Die Klasse "SelectionServiceImpl" trifft Routing-Entscheidungen. Wenn der Client Failover-Entscheidungen trifft, können Sie diese Klasse verwenden, um die aus den Entscheidungen resultierenden Aktionen zu überwachen.

ObjectGridImpl
: Die Klasse "ObjectGridImpl" hat die Methode "getSession", die Sie überwachen können, um die Anzahl der Anforderungen an diese Methode zu erhalten.

eXtreme Scale mit Hyperic HQ überwachen

Hyperic HQ ist eine Überwachungslösung eines anderen Anbieters, die kostenlos als Open-Source-Lösung oder als Unternehmensprodukt verfügbar ist. WebSphere eXtreme Scale enthält ein Plug-in, mit dem Hyperic-HQ-Agenten eXtreme-Scale-Containerserver erkennen und Statistiken mit Hilfe von eXtreme-Scale-Management-Beans berichten und zusammenfassen können. Sie können Hyperic HQ verwenden, um eigenständige Implementierungen von eXtreme Scale zu überwachen.

Vorbereitende Schritte

- Die folgenden Anweisungen gelten für Hyperic Version 4.0. Wenn Sie eine neuere Version von Hyperic haben, schlagen Sie in der Hyperic-Dokumentation Informationen wie Pfadnamen und Informationen zum Starten von Agenten und Servern nach.
- Laden Sie Hyperic-Server- und -Agenteninstallationen herunter. Eine Serverinstallation muss aktiv sein. Um alle eXtreme-Scale-Server zu erkennen, muss ein Hyperic-Agent auf jeder Maschine ausgeführt werden, auf dem ein eXtreme-Sca-

le-Server ausgeführt wird. Downloadinformationen und Dokumentationsunterstützung finden Sie auf der Website von Hyperic.

- Sie müssen Zugriff auf die Dateien `objectgrid-plugin.xml` und `hqplugin.jar` haben. Diese Dateien befinden sich im Verzeichnis `ObjectGrid-Stammverzeichnis/hyperic/etc`.

Informationen zu diesem Vorgang

Durch die Integration der Überwachungssoftware Hyperic HQ in eXtreme Scale können Sie Metriken zur Leistung Ihrer Umgebung grafisch überwachen und anzeigen. Sie konfigurieren diese Integration über eine Plug-in-Implementierung in jedem Agenten.

Vorgehensweise

1. Starten Sie Ihre eXtreme-Scale-Server. Das Hyperic-Plug-in prüft die lokalen Prozesse, um eine Verbindung zu den Java Virtual Machines herzustellen, in denen eXtreme Scale ausgeführt wird. Für eine ordnungsgemäße Verbindungsherstellung zu den Java Virtual Machines muss jeder Server mit der Option **-jmxServicePort** gestartet werden. Informationen zum Starten von Servern mit der Option **-jmxServicePort** finden Sie im Abschnitt „Script "startOgServer"" auf Seite 340.
2. Kopieren Sie die Datei `extremescale-plugin.xml` und die Datei `wxshyperic.jar` in die entsprechenden Server- und Agenten-Plug-in-Verzeichnisse in Ihrer Hyperic-Konfiguration. Zum Integrieren von Hyperic müssen die Agenten- und die Serverinstallationen Zugriff auf das Plug-in und die JAR-Dateien haben. Obwohl der Server Konfigurationen dynamisch wechseln kann, müssen Sie die Integration vor dem Starten von Agenten durchführen.
   a. Kopieren Sie die Datei `extremescale-plugin.xml` in das Serververzeichnis `plugin` an der folgenden Position:
      `hyperic_home/server_home/hq-engine/server/default/deploy/hq.ear/hq-plugins`
   b. Kopieren Sie die Datei `extremescale-plugin.xml` in das Agentenverzeichnis `plugin` an der folgenden Position:
      `agent_home/bundles/gent-4.0.2-939/pdk/plugins`
   c. Kopieren Sie die Datei `wshyperic.jar` in das Agentenverzeichnis `lib` an der folgenden Position:
      `agent_home/bundles/gent-4.0.2-939/pdk/lib`
3. Konfigurieren Sie den Agenten. Die Datei `agent.properties` dient als Konfigurationspunkt für die Agentenlaufzeitumgebung. Diese Datei ist im Verzeichnis *`agent_home`*`/conf` enthalten. Die folgenden Schlüssel sind optional, aber wichtig für das eXtreme-Scale-Plug-in:
   - `autoinventory.defaultScan.interval.millis=<`*`Millisekunden`*`>`

     Legt das Intervall (in Millisekunden) fest, in dem die Erkennungsoperationen des Agenten ausgeführt werden.
   - `log4j.logger.org.hyperic.hq.plugin.extremescale.XSServerDetector=DEBUG`

     Aktiviert ausführliche Debug-Anweisungen vom eXtreme-Scale-Plug-in.
   - `username=<`*`Benutzername`*`>`: Legt den JMX-Benutzernamen (Java Management Extensions) fest, wenn die Sicherheit aktiviert ist.
   - `password=<`*`Kennwort`*`>`: Legt das JMX-Kennwort fest, wenn die Sicherheit aktiviert ist.

- `sslEnabled=<true|false>`: Teilt dem Plug-in mit, ob Secure Sockets Layer (SSL) verwendet werden soll. Der Standardwert ist `false`.
- `trustPath=<Pfad>`: Legt den Sicherheitspfad für die SSL-Verbindung fest.
- `trustType=<Typ>`: Legt den Sicherheitstyp für die SSL-Verbindung fest.
- `trustPass=<Kennwort>`: Legt das Sicherheitskennwort für die SSL-Verbindung fest.

4. Starten Sie die Agentenerkennung. Die Hyperic-Agenten senden Erkennungs- und Metrikinformationen an den Server. Verwenden Sie den Server, um Datensichten anzupassen und logische Bestandsobjekt zu gruppieren, um hilfreiche Informationen zu erhalten. Wenn der Server verfügbar ist, müssen Sie das Startscript ausführen oder den Windows-Dienst für den Agenten starten:
   - Linux `agent_home/bin/hq-agent.sh start`
   - Windows Starten Sie den Agenten mit dem Windows-Dienst.

   Nachdem Sie den Agenten gestartet haben, werden die Server erkannt und Gruppen konfiguriert. Sie können sich an der Serverkonsole anmelden und die Ressourcen auswählen, die der Bestandsdatenbank für den Server hinzugefügt werden sollen. Die Serverkonsole ist standardmäßig über den folgenden URL erreichbar: `http://<Hostname_des_Servers>:7080/`
5. Statistiken müssen aktiviert werden, damit Hyperic statistische Daten erfasst.

   Verwenden Sie die Steueraktion **SetStatsSpec** in der Hyperic-Konsole für eXtreme Scale. Navigieren Sie zur Ressource, und verwenden Sie anschließend das Dropdown-Menü **Control Action** auf der Registerkarte **Control**, um `SetStatsSpec` mit der Einstellung `ALL=enabled` im Textfeld **Control Arguments** einzugeben.

   Katalogserver werden von dem Filtersatz in der Hyperic-Konsole nicht erkannt. Lesen Sie die Informationen zur Eigenschaft **statsSpec** im Abschnitt „Servereigenschaftendatei” auf Seite 188, mit der Statistiken beim Containerstart aktiviert werden. Verschiedene Optionen zum Aktivieren von Statistiken sind in „Überwachung mit Managed Beans (MBeans)” auf Seite 406 und „Überwachung mit dem Musterdienstprogramm "xsAdmin"” auf Seite 391 beschrieben.
6. Überwachen Sie Server mit der Hyperic-Konsole. Nachdem die Server dem Bestandsmodell hinzugefügt wurden, sind ihre Services nicht mehr erforderlich.
   - **Dashboard-Sicht**: Wenn Sie die Ressourcenerkennungsereignisse anzeigen möchten, müssen Sie sich bei der Haupt-Dashboard-Sicht anmelden. Die Dashboard-Sicht ist eine generische Sicht, die als Message Center dient, das Sie anpassen können. Sie können Graphen oder Bestandsobjekte in dieses Haupt-Dashboard exportieren.
   - **Ressourcensicht:** Sie können das gesamte Bestandsmodell über diese Seite abfragen und anzeigen. Nachdem Sie die Server hinzugefügt haben, wird jeder eXtreme-Scale-Server ordnungsgemäß beschriftet im Abschnitt mit den Servern aufgelistet. Sie können auf die einzelnen Server klicken, um die Basismetriken anzuzeigen.
7. Zeigen Sie den gesamten Serverbestand auf der Seite mit der Ressourcensicht an. Auf dieser Seite können Sie mehrere ObjectGrid-Server auswählen und gruppieren. Nach der Gruppierung von Ressourcen können die gemeinsamen Metriken der jeweiligen Gruppe in Graphen dargestellt werden, um Überschneidungen und Unterschiede zwischen den Gruppen-Membern anzuzeigen. Um eine Überschneidung anzuzeigen, wählen Sie die Metrik in der Anzeige Ihrer Servergruppe aus. Die Metrik wird dann im Diagrammbereich angezeigt. Zum Anzeigen einer Überschneidung für alle Gruppen-Member klicken Sie auf

den unterstrichenen Metriknamen. Sie können alle Diagramme, Knotensichten und Vergleichsüberschneidungen über das Menü **Tools** in das Haupt-Dashboard exportieren.

Kapitel 10. Optimierung und Leistung

Prüfliste für die Betriebsbereitschaft

Verwenden Sie die Prüfliste für die Betriebsbereitschaft, um Ihre Umgebung für die Implementierung von WebSphere eXtreme Scale vorzubereiten.

Tabelle 27. Prüfliste für die Betriebsbereitschaft

Prüflistenpunkt	Weiterführende Informationen

Wenn Sie AIX verwenden, optimieren Sie die folgenden Betriebssystemeinstellungen:
 	
TCP_KEEPINTVL
Die Einstellung TCP_KEEPINTVL gehört zu einem Socket-Keepalive-Protokoll, das die Erkennung von Netzausfällen ermöglicht. Die Eigenschaft gibt das Intervall an, in dem Pakete zum Validieren der Verbindung gesendet werden. Wenn Sie WebSphere eXtreme Scale verwenden, setzen Sie die Eigenschaft auf `10`. Zum Überprüfen der aktuellen Einstellung führen Sie den folgenden Befehl aus:
`# no –o tcp_keepintvl`
 	
Führen Sie den folgenden Befehl aus, um die aktuelle Einstellung zu ändern:
`# no –o tcp_keepintvl=10`
 	
Der Wert für die Einstellung TCP_KEEPINTVL wird in Halbsekunden angegeben.
 	
TCP_KEEPINIT
Die Einstellung TCP_KEEPINIT gehört zu einem Socket-Keepalive-Protokoll, das die Erkennung von Netzausfällen ermöglicht. Die Eigenschaft gibt das Anfangszeitlimit für die TCP-Verbindung an. Wenn Sie WebSphere eXtreme Scale verwenden, setzen Sie die Eigenschaft auf `40`. Zum Überprüfen der aktuellen Einstellung führen Sie den folgenden Befehl aus:
`# no –o tcp_keepinit`
 	
Führen Sie den folgenden Befehl aus, um die aktuelle Einstellung zu ändern:
`# no –o tcp_keepinit=40`
 	
Der Wert für die Einstellung TCP_KEEPINIT wird in Halbsekunden angegeben.
	• Informationen zur Optimierung von AIX finden Sie unter "AIX-Systeme optimieren".

Aktualisieren Sie die Datei `orb.properties`, um das Transportverhalten des Grids zu ändern. Sie finden die Datei `orb.properties` im Verzeichnis `java/jre/lib`.	„ORB-Eigenschaftendatei” auf Seite 197

Tabelle 27. Prüfliste für die Betriebsbereitschaft (Forts.)

Prüflistenpunkt	Weiterführende Informationen
Verwenden Sie Parameter im Script "startOgServer", insbesondere die folgenden: • Legen Sie die Einstellungen für den Heap-Speicher mit dem Parameter **-jvmArgs** fest. • Legen Sie den Anwendungsklassenpfad und Anwendungseigenschaften mit dem Parameter **-jvmArgs** fest. • Setzen Sie den Parameter **-jvmArgs** für die Konfiguration der Agentenüberwachung. **Porteinstellungen** WebSphere eXtreme Scale muss für einige Transporte Kommunikationsports öffnen. Diese Ports werden alle dynamisch definiert. Wenn jedoch eine Firewall zwischen den Containern verwendet wird, müssen Sie die Ports angeben. Verwenden Sie die folgenden Portinformationen: **Listener-Port** Sie können das Argument **-listenerPort** verwenden, um den Port anzugeben, der für die Kommunikation zwischen Prozessen verwendet wird. **Stammgruppenport** Sie können das Argument **-haManagerPort** verwenden, um den Port anzugeben, der für die Fehlererkennung verwendet wird. Dieses Argument ist mit "peerPort" identisch. Beachten Sie, dass Stammgruppen nicht zonenübergreifend kommunizieren müssen. Deshalb müssen Sie diesen Port unter Umständen nicht setzen, wenn die Firewall für alle Member einer einzigen Zone offen ist. **JMX-Serviceport** Sie können das Argument **-JMXServicePort** verwenden, um den Port anzugeben, den der JMX-Service verwenden soll. **SSL-Port** Wenn Sie `-Dcom.ibm.CSI.SSLPort=1234` als Argument mit **-jvmArgs** angeben, wird der SSL-Port auf `1234` gesetzt. Der SSL-Port ist der sichere Port-Peer zum Listener-Port. **Clientport** Wird nur im Katalogservice verwendet. Sie können diesen Wert mit dem Argument **-catalogServiceEndPoints** angeben. Der Wert für diesen Parameter muss im folgenden Format angegeben werden: `Servername:Hostname:Clientport:Peerport`	„Script "startOgServer"" auf Seite 340
Stellen Sie sicher, dass die Sicherheitseinstellungen ordnungsgemäß konfiguriert sind: • Transport (SSL) • Anwendung (Authentifizierung und Berechtigung) Zum Überprüfen Ihrer Sicherheitseinstellungen können Sie versuchen, einen zerstörerischen Client zu verwenden, der eine Verbindung zu Ihrer Konfiguration herstellt. Wenn beispielsweise die Einstellung "SSL-Required" konfiguriert ist, sollte ein Client, der die Einstellung TCP_IP hat, oder ein Client mit dem falschen Truststore nicht in der Lage sein, eine Verbindung zum Server herzustellen. Wenn eine Authentifizierung erforderlich ist, sollte ein Client ohne Berechtigungsnachweis (z. B. ohne Benutzer-ID und Kennwort) nicht in der Lage sein, eine Verbindung zum Server herzustellen. Wenn die Berechtigung zwingend erforderlich ist, sollte einem Client ohne Zugriffsberechtigung der Zugriff auf die Serverressourcen nicht erteilt werden.	„Sicherheitsintegration mit externen Providern" auf Seite 375
Legen Sie fest, wie die Umgebung überwacht werden soll. • xsAdmin – Die JMX-Ports der Katalogserver müssen für das Tool "XSAdmin" sichtbar sein. Auch die Containerport müssen für einige Befehle, die Informationen von den Containern erfassen, zugänglich sein. • Sie können zwischen den folgenden Überwachungstools wählen: – Tivoli Enterprise Monitoring Agent – CA Wily Introscope – Hyperic HQ	• „Überwachung mit dem Musterdienstprogramm "xsAdmin"" auf Seite 391 • „JMX-Sicherheit (Java Management Extensions)" auf Seite 372 • „Überwachung mit IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale" auf Seite 414 • „eXtreme Scale mit Hyperic HQ überwachen" auf Seite 424 • „eXtreme-Scale-Anwendungen mit CA Wily Introscope überwachen" auf Seite 420

Optimierung von Betriebssystem und Netz

Eine Netzoptimierung kann durch Änderung der Verbindungseinstellung Verzögerungen im TCP-Stack (Transmission Control Protocol) reduzieren und den Durchsatz durch Änderung der TCP-Puffer verbessern.

Betriebssysteme

Ein Windows-System erfordert die geringste Optimierung, ein Solaris-System die meiste. Die folgenden Informationen gelten für jedes angegebene System und können die Leistung von WebSphere eXtreme Scale verbessern. Sie müssen die Optimierung Ihrem Netz und Ihrer Anwendungslast entsprechend optimieren.

Windows

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\
Tcpip\Parameters
MaxFreeTcbs = dword:00011940
MaxHashTableSize = dword:00010000
MaxUserPort = dword:0000fffe
TcpTimedWaitDelay = dword:0000001e
```

Solaris

```
ndd -set /dev/tcp tcp_time_wait_interval 60000
fndd -set /dev/tcp tcp_keepalive_interval 15000
ndd -set /dev/tcp tcp_fin_wait_2_flush_interval 67500
ndd -set /dev/tcp tcp_conn_req_max_q 16384
ndd -set /dev/tcp tcp_conn_req_max_q0 16384
ndd -set /dev/tcp tcp_xmit_hiwat 400000
ndd -set /dev/tcp tcp_recv_hiwat 400000
ndd -set /dev/tcp tcp_cwnd_max 2097152
ndd -set /dev/tcp tcp_ip_abort_interval 20000
ndd -set /dev/tcp tcp_rexmit_interval_initial 4000
ndd -set /dev/tcp tcp_rexmit_interval_max 10000
ndd -set /dev/tcp tcp_rexmit_interval_min 3000
ndd -set /dev/tcp tcp_max_buf 4194304
```

AIX

```
/usr/sbin/no -o tcp_sendspace=65536
/usr/sbin/no -o tcp_recvspace=65536
/usr/sbin/no -o udp_sendspace=65536
/usr/sbin/no -o udp_recvspace=65536
/usr/sbin/no -o somaxconn=10000
/usr/sbin/no -o tcp_nodelayack=1
/usr/sbin/no –o tcp_keepinit=40
/usr/sbin/no –o tcp_keepintvl=10
```

LINUX

```
sysctl -w net.ipv4.tcp_timestamps=0
sysctl -w net.ipv4.tcp_tw_reuse=1
sysctl -w net.ipv4.tcp_tw_recycle=1
sysctl -w net.ipv4.tcp_fin_timeout=30
sysctl -w net.ipv4.tcp_keepalive_time=1800
sysctl -w net.ipv4.tcp_rmem="4096 87380 8388608"
sysctl -w net.ipv4.tcp_wmem="4096 87380 8388608"
sysctl -w net.ipv4.tcp_max_syn_backlog=4096
```

HP-UX

```
ndd -set /dev/tcp tcp_ip_abort_cinterval 20000
```

Netzports planen

WebSphere eXtreme Scale ist ein verteilter Cache, der das Öffnen von Ports für die Kommunikation mit dem ORB (Object Request Broker) und dem TCP-Stack (Transmission Control Protocol) zwischen Java Virtual Machines und anderen Maschinen voraussetzt. Sie sollten Ihre Ports planen und steuern, insbesondere in einer Firewall-Umgebung, z. B., wenn Sie Katalogservices und Container an mehreren Ports verwenden.

Katalogservicedomäne

Eine Katalogservicedomäne erfordert die Definition der folgenden Ports:

peerPort
Gibt den Port für den High-Availability Manager an, über den Peer-Katalogserver über einen TCP-Stack miteinander kommunizieren.

clientPort
Gibt den Port für den Zugriff auf Katalogservicedaten für Katalogserver an.

JMXServicePort
Gibt an, welchen Port der JMX-Server (Java Management Extensions) verwenden soll.

listenerPort
Definiert den ORB-Listener-Port, den Container und Clients für die Kommunikation mit dem Katalogserver über den ORB verwenden.

Wie Sie diese Ports definieren, richtet sich danach, ob Sie den eigenständigen Modus verwenden oder ob Sie die Katalogserver von eXtreme Scale in einer Umgebung von WebSphere Application Server starten:

- **Für den eigenständigen Modus:**

  Verwenden Sie den Befehl "startOgServer", um die zuvor mit der Option aufgelisteten Ports im eigenständigen Modus anzugeben, wie im folgenden Beispiel gezeigt wird:

  ```
  –catalogServiceEndPoints cs1:host1:clientPort:peerPort, cs2:host2:clientPort:peerPort,
  cs3:host3:clientPort:peerPort -listenerPort <ORB-Port> -JMXServicePort <JMX-Port>
  ```

  Weitere Informationen zum Starten der Katalogserver im eigenständigen Modus finden Sie unter „Katalogservice in einer eigenständigen Umgebung starten” auf Seite 335.

- **Für eine Umgebung mit WebSphere Application Server:**

  Sie können eine Katalogservicedomäne in der Administrationskonsole definieren. Weitere Informationen finden Sie unter „Katalogservicedomänen in WebSphere Application Server erstellen” auf Seite 350.

Containerserver

Die Containerserver von WebSphere eXtreme Scale erfordern für den Betrieb ebenfalls verschiedene Ports. Standardmäßig generiert der Containerserver von eXtreme Scale seinen HA-Manager-Port und seinen ORB-Listener-Port automatisch mit dynamischen Ports. Da es in der Firewall-Umgebung empfehlenswert ist, Ports zu planen und zu steuern, werden Optionen bereitgestellt, um Containerserver von eXtreme Scale, wie im folgenden Beispiel gezeigt, mit dem angegebenen HA-Manager-Port und dem angegebenen ORB-Listener-Port mit einer Option im Befehl "startOgServer" zu starten:

```
-HaManagerPort <Peer-Port>
-listenerPort <ORB-Port>
```

Eine ordnungsgemäße Planung der Portsteuerung ist vorteilhaft, aber die Planung und Verwaltung dieser Ports kann sich natürlich schwierig gestalten, wenn Hunderte von Java Virtual Machines in einer Maschine gestartet werden. Jeder Portkonflikt führt zu einem Fehler beim Serverstart.

Wenn die Sicherheit aktiviert ist, muss der vorherigen Portliste ein SSL-Port (Secure Sockets Layer) hinzugefügt werden. Wenn Sie `Dcom.ibm.CSI.SSLPort=<sslPort>` als Argument für **-jvmArgs** verwenden, wird der SSL-Port auf `<sslPort>` gesetzt. Als

Unterstützung bei der Portplanung sollten Sie die Informationen zu den Sicherheitseinstellungen für eXtreme Scale lesen.

ORB-Eigenschaften und Dateideskriptoreinstellungen

Die Optimierungshinweise beziehen sich unter anderem auf die ORB-Eigenschaften (Object Request Broker) und die Dateideskriptoreinstellungen.

ORB-Eigenschaften

Der ORB (Object Request Broker) wird von WebSphere eXtreme Scale für die Kommunikation über einen TCP-Stack verwendet. Sie finden die erforderliche Datei `orb.properties` im Verzeichnis `java/jre/lib`. Für Lastspitzen mit großen Objekten aktivieren Sie die ORB-Fragmentierung mit der folgenden Einstellung:

```
com.ibm.CORBA.FragmentSize=<richtige Größe>
```

Mit der folgenden Einstellung können Sie das Anwachsen des Thread-Pools verhindern:

```
com.ibm.CORBA.ThreadPool.IsGrowable=false
```

Mit den folgenden Einstellungen können Sie angemessene Zeitlimits festlegen, um die Ausführung zu vieler Threads in einer abnormalen Situation zu verhindern:

- com.ibm.CORBA.RequestTimeout
- com.ibm.CORBA.ConnectTimeout
- com.ibm.CORBA.FragmentTimeout

Dateideskriptor

Auf UNIX- und Linux-Systemen ist die zulässige Anzahl offener Dateien pro Prozess beschränkt. Das Betriebssystem gibt die zulässige Anzahl offener Dateien an. Wenn dieser Wert zu klein ist, tritt ein Fehler bei der Speicherzuordnung unter AIX auf, und es wird protokolliert, dass zu viele Dateien offen sind.

Setzen Sie diese Einstellung im Terminalfenster auf dem UNIX-System auf einen höheren Wert als den Systemstandardwert. Für große SMP-Maschinen mit Klonen legen Sie den Wert für eine uneingeschränkte Anzahl offener Dateien fest.

Für AIX-Konfiguration setzen Sie diese Einstellung mit dem Befehl `ulimit -n -1` auf den Wert `-1` (uneingeschränkt).

Für Solaris-Konfigurationen setzen Sie diese Einstellung mit dem Befehl `ulimit -n 16384` auf den Wert `16384`.

Zum Anzeigen des aktuellen Werts verwenden Sie den Befehl `ulimit –a`.

NIO mit dem ORB

Momentan können Sie mit NIO (Non-Blocking Input/Output, Nicht blockierende Ein-/Ausgabe) oder ChannelFramework in eigenständigen eXtreme-Scale-Szenarien arbeiten. Um NIO im IBM ORB zu verwenden, müssen Sie den Transportmodus des IBM ORB auf ChannelFramework setzen. Standardmäßig wird der IBM ORB im Modus "Pluggable" ausgeführt. WebSphere eXtreme Scale stellt eine Servereigenschaft bereit, um den Transportmodus auf "ChannelFramework" zu setzen.

Wichtig:

WebSphere Application Server unterstützt den Modus "Pluggable" nur in den aktuellen Releases. Wenn WebSphere eXtreme Scale mit WebSphere Application Server integriert ausgeführt wird, muss der Modus "Pluggable" verwendet werden. Da WebSphere eXtreme Scale auch die SSL-/Transportsicherheit von WebSphere Application Server verwendet, wird deshalb momentan auch nur der Modus "Pluggable" unterstützt.

Verwendung von NIO

Es folgt ein kleiner Abschnitt zur Einführung in das Konzept.

NIO oder ChannelFramework im ORB aktivieren

WebSphere eXtreme Scale stellt eine Servereigenschaft bereit, mit der "TransportMode" auf "ChannelFramework" gesetzt werden kann, um NIO (Non-blocking I/O, nicht blockierende Ein-/Ausgabe) im ORB zu aktivieren.

Vorbereitende Schritte

Suchen Sie die vorhandene Servereigenschaftendatei, oder erstellen Sie eine Servereigenschaftendatei. Sie können ChannelFramework im Katalogservice und in den Containerservern aktivieren. Weitere Einzelheiten finden Sie in der Beschreibung der Servereigenschaftendatei.

Informationen zu diesem Vorgang

Momentan können Sie mit NIO (Non-Blocking Input/Output, Nicht blockierende Ein-/Ausgabe) oder ChannelFramework in eigenständigen eXtreme-Scale-Szenarien arbeiten. Um NIO im IBM ORB zu verwenden, müssen Sie den Transportmodus des IBM ORB auf ChannelFramework setzen. Standardmäßig wird der IBM ORB im Modus "Pluggable" ausgeführt. WebSphere eXtreme Scale stellt eine Servereigenschaft bereit, um den Transportmodus auf "ChannelFramework" zu setzen.

Wichtig:

WebSphere Application Server unterstützt den Modus "Pluggable" nur in den aktuellen Releases. Wenn WebSphere eXtreme Scale mit WebSphere Application Server integriert ausgeführt wird, muss der Modus "Pluggable" verwendet werden. Da WebSphere eXtreme Scale auch die SSL-/Transportsicherheit von WebSphere Application Server verwendet, wird deshalb momentan auch nur der Modus "Pluggable" unterstützt.

Vorgehensweise

1. Fügen Sie die Eigenschaft "enableChannelFramework=true" Ihrer Servereigenschaftendatei hinzu.
2. Stellen Sie sicher, dass die Servereigenschaftendatei der ORB-Eigenschaftendatei nicht widerspricht.

   Wenn die Servereigenschaftendatei den Transportmodus "ChannelFramework" aktiviert, aber das Attribut "TransportMode" in der Datei "orb.properties" auf "Pluggable" gesetzt ist, überschreibt der Server die Einstellung in der Datei "orb.properties" nicht. Es wird eine Warnung im Protokoll aufgezeichnet, in der darauf hingewiesen wird, dass es zwei Einstellungen gibt. Damit die Eigenschaft "enableChannelFramework=true" wirksam wird, müssen Sie die Eigenschaften anpassen, die anzeigen, dass "TransportMode" auf "Pluggable" gesetzt ist. Ändern Sie "com.ibm.CORBA.TransportMode=Pluggable" in "ChannelFramework", oder entfernen Sie die Eigenschaft.

3. Stellen Sie die aktualisierte Eigenschaftendatei beim Start des Katalogservice bzw. Containerservers bereit. Weitere Einzelheiten zur Verwendung der Servereigenschaften zum Starten eines Servers finden Sie unter Servereigenschaftendatei.

Ergebnisse

Wenn ein Katalogservice oder Containerserver den Transportmodus "channelFramework" verwendet, wird die folgende Nachricht im Protokoll ausgegeben.

```
CWOBJ0052I: Die IBM ORB-Eigenschaft TransportMode wurde auf ChannelFramework gesetzt.
```

Wenn Sie die folgende Nachricht im Protokoll sehen, überprüfen Sie, wie zuvor beschrieben, Ihre ORB-Eigenschaften.

```
CWOBJ0055W: Die IBM ORB-Eigenschaft TransportMode wurde in der Servereigenschaftendatei auf Channe
gesetzt, aber die vorhandene Datei orb.properties enthält bereits eine Einstellung für TransportMo
Die TransportMode-Einstellung wird nicht überschrieben.
```

Wenn Sie ChannelFramework aktivieren, ist 512 der Maximalwert für ServerSocketQueueDepth. Wenn die Einstellung von ServerSocketQueueDepth in der Datei "orb.properties" höher als 512 ist, setzt der Server ServerSocketQueueDepth in der Datei "orb.properties" automatisch auf 512 und informiert Sie in Informationsnachricht im Protokoll darüber. Es ist keine Aktion erforderlich.

```
CWOBJ0053I: Die IBM ORB-Eigenschaft ServerSocketQueueDepth wurde auf 512
gesetzt, damit eine ordnungsgemäße Ausführung mit der TransportMode-Einstellung ChannelFramework m
```

JVM-Optimierung für WebSphere eXtreme Scale

Durch die Optimierung der Java Virtual Machine (JVM) können erhebliche Verbesserungen in der Implementierung von WebSphere eXtreme Scale erzielt werden.

In den meisten Fällen erfordert WebSphere eXtreme Scale nur wenige bzw. keine speziellen JVM-Einstellungen. Wenn Sie sehr viele Objekte haben, die in WebSphere eXtreme Scale gespeichert werden, passen Sie die Größe des Heap-Speichers entsprechend an, um Speicherengpässe zu vermeiden.

Plattformen

Leistungstests wurden hauptsächlich auf AIX- (32 Wege), Linux- (4 Wege) und Windows-Maschinen (8 Wege) durchgeführt. Auf High-End-AIX-Maschinen konnten Multithread-Szenarien mit hoher Last realisiert und Konfliktpunkte identifiziert und ausgeräumt werden.

ORB-Anforderungen (Object Request Broker)

IBM SDK enthält eine IBM ORB-Implementierung, die mit WebSphere Application Server und WebSphere eXtreme Scale getestet wurde. Zur Vereinfachung des Unterstützungsprozesses sollten Sie eine von IBM bereitgestellte JVM verwenden. Andere JVM-Implementierungen verwenden einen anderen ORB. Der ohne Vorbereitungs- oder Anpassungsaufwand einsatzfähige IBM ORB wird nur mit JVMs von IBM bereitgestellt. WebSphere eXtreme Scale erfordert für den Betrieb einen funktionierenden ORB. Sie können WebSphere eXtreme Scale zwar auch mit ORBs anderer Anbieter verwenden, aber wenn ein Problem im ORB auftritt, müssen Sie sich für Unterstützung an den ORB-Anbieter wenden. Die IBM ORB-Implementierung ist mit JVMs anderer Anbieter kompatibel und kann bei Bedarf ersetzt werden.

Garbage-Collection

WebSphere eXtreme Scale erstellt temporäre Objekte für jede Transaktion, wie z. B. Anforderungs- und Antwortobjekte und eine Protokollfolge. Da sich diese Objekte auf die Effizienz der Garbage-Collection auswirken, ist die Optimierung der Garbage-Collection kritisch.

Verwenden Sie für die IBM Virtual Machine for Java den Collector "optavgpause" in Szenarien mit einer hohen Aktualisierungsrate (100 % geänderter Transaktionseinträge). Der Collector "gencon" funktioniert in Szenarien, in denen die Daten nur relativ selten aktualisiert werden (10 % der Zeit oder weniger), viel besser als der Collector "optavgpause". Experimentieren Sie mit beiden Collector, um festzustellen, welcher sich besser in Ihrem Szenario eignet. Wenn Sie ein Leistungsproblem erkennen, aktivieren Sie die ausführliche Garbage-Collection, um die für die Garbage-Collection aufgebrachte Zeit in Prozent zu überprüfen. Es wurden Szenarien gefunden, in denen 80 % der Zeit für die Garbage-Collection aufgebracht wurde, bis das Problem durch Optimierung behoben wurde.

Weitere Informationen zum Konfigurieren der Garbage-Collection finden Sie unter IBM Virtual Machine for Java optimieren.

JVM-Leistung

WebSphere eXtreme Scale kann unter verschiedenen Versionen von Java 2 Platform, Standard Edition (J2SE) ausgeführt werden. WebSphere eXtreme Scale Version 6.1 unterstützt J2SE Version 1.4.2 und höher. Zur Steigerung der Entwicklerproduktivität und der Leistung verwenden Sie J2SE 5 oder höher, um die Vorteile von Annotationen und verbesserter Garbage-Collection zu nutzen. WebSphere eXtreme Scale kann in 32-Bit- und 64-Bit-JVMs ausgeführt werden.

WebSphere eXtreme Scale wurde mit einem Teil der verfügbaren virtuellen Maschinen getestet, aber die Liste der unterstützten JVMs ist nicht exklusiv. Sie können WebSphere eXtreme Scale in jeder Version 1.4.2 oder höher ausführen, aber wenn ein Problem in der JVM auftritt, müssen Sie sich für Unterstützung an den jeweiligen JVM-Anbieter wenden. Verwenden Sie, sofern möglich, die JVM aus der WebSphere-Laufzeitumgebung auf jeder Plattform, die von WebSphere Application Server unterstützt wird.

In den meisten Szenarien, in denen WebSphere eXtreme Scale verwendet wird, bietet Java Platform Standard Edition 6 der JVM eine bessere Leistung als Edition 5 oder 1.4. Die Leistung von Java 2 Platform, Standard Edition Version 1.4 ist insbesondere in Szenarien, in denen der Collector "gencon" verwendet wird, sehr schwach. Java Platform Standard Edition 5 bietet eine gute Leistung, aber Java Platform, Standard Edition 6 eine bessere.

Große Heap-Speicher

Wenn die Anwendung ein großes Datenvolumen für jede Partition verwalten muss, kann die Garbage-Collection ein entscheidender Faktor sein. In einem Szenario, in dem hauptsächlich Leseoperationen durchgeführt werden, ist die Leistung selbst bei sehr großen Heap-Speichern (20 GB und mehr) gut, wenn eine Garbage-Collection nach Objektalter verwendet wird. Wenn der Heap-Speicher für die permanenten Objekte gefüllt ist, ist jedoch eine Pause zu bemerken, die proportional zur Größe des Live-Heap-Speichers und der Anzahl der Prozessoren in der Maschine ist. Diese Pause kann auf kleineren Maschinen mit großen Heap-Speichern sehr lang sein.

WebSphere eXtreme Scale unterstützt WebSphere Real Time Java. Mit WebSphere Real Time Java sind die Transaktionsverarbeitungsantworten für WebSphere eXtreme Scale konsistenter und vorhersehbar, und die Auswirkungen der Garbage-Collection und der Thread-Planung ist erheblich geringer. Die Auswirkungen werden so weit reduziert, dass die Standardabweichung bei den Antwortzeiten weniger als 10 % als beim regulären Java beträgt.

Weitere Informationen finden Sie im Abschnitt „WebSphere Real Time verwenden" auf Seite 444.

Thread-Anzahl

Die Thread-Anzahl ist von verschiedenen Faktoren abhängig. Die Anzahl der Threads, die von einem einzelnen Shard verwaltet werden können, ist begrenzt. Ein Shard ist eine Instanz einer Partition. Ein Shard kann ein primäres Shard oder ein Replikat-Shard sein. Je mehr Shards jede JVM enthält, desto mehr Threads und mehr gemeinsame Datenzugriffspfade sind möglich. Jedes Shard ist so nebenläufig wie möglich, aber nichtsdestotrotz gibt es einen Grenzwert.

Optimierung von JVMs

Sie müssen bestimmte Aspekte der JVM-Optimierung (Java Virtual Machine) berücksichtigen, um die beste Leistung mit WebSphere eXtreme Scale zu erzielen.

Empfohlen werden 1- bis 2-Gb-Heap-Speicher mit einer JVM pro 4 Kernen. Die Größe der Heap-Speicher richtet sich nach der Art der Objekte, die in den Servern gespeichert werden. Eine diesbezügliche Erläuterung finden Sie weiter hinten in diesem Dokument.

Empfehlungen zur Größe von Heap-Speichern und zur Garbage-Collection

Die optimale Größe der Heap-Speicher ist von drei Faktoren abhängig:

1. Anzahl der Liveobjekte im Heap-Speicher,
2. Komplexität der Liveobjekte im Heap-Speicher,
3. Anzahl verfügbarer Kerne für die JVM.

Eine Anwendung, die beispielsweise 10-KB-Feldgruppen speichert, kann einen viel größeren Heap-Speicher haben als eine Anwendung, die komplexe POJO-Graphen verwendet.

Alle modernen JVMs verwenden Algorithmen für parallele Garbage-Collection, d. h., durch den Einsatz weiterer Kerne können die Pausen zwischen den Garbage-Collections reduziert werden. Das bedeutet also, dass die Garbage-Collection auf Maschinen mit 8 Kernen schneller ist als auf einer Maschine mit 4 Kernen.

Realspeicherbelegung versus Heap-Speicherspezifikation

Eine JVM mit einem 1-Gb-Heap-Speicher belegt ungefähr 1,3 Gb Realspeicher. Unter Laborbedingungen wurde festgestellt, dass es nicht möglich ist, zehn 1-Gb-JVMs auf einer Maschine mit 16 Gb Arbeitsspeicher auszuführen. Sobald die JVM eine Belegung von 800 MB erreicht, beginnt sie mit dem Paging.

Garbage-Collection

Verwenden Sie für IBM JVMs den Collector "avgoptpause" in Szenarien mit einer hohen Aktualisierungsrate (100 % geänderter Transaktionseinträge). Der Collector "gencon" funktioniert in Szenarien, in denen die Daten nur relativ selten aktualisiert werden (10 % der Zeit oder weniger), viel besser als der Collector "avgoptpause". Experimentieren Sie mit beiden Collector-Typen, um festzustellen, welcher sich besser in Ihrem Szenario eignet. Wenn Sie ein Leistungsproblem erkennen, aktivieren Sie die ausführliche Garbage-Collection, um die für die Garbage-Collection aufgebrachte Zeit in Prozent zu überprüfen. Es wurden Szenarien gefunden, in denen 80 % der Zeit für die Garbage-Collection aufgebracht wurde, bis das Problem durch Optimierung behoben wurde.

JVM-Leistung

WebSphere eXtreme Scale kann unter verschiedenen Versionen von Java 2 Platform, Standard Edition (J2SE) ausgeführt werden. ObjectGrid Version 6.1 unterstützt J2SE Version 1.4.2 und höher. Zur Steigerung der Entwicklerproduktivität und der Leistung verwenden Sie J2SE 5 oder höher, um die Vorteile von Annotationen und verbesserter Garbage-Collection zu nutzen. ObjectGrid funktioniert in 32-Bit- und 64-Bit-JVMs.

Clients von ObjectGrid Version 6.0.2 können eine Verbindung zu einem Grid der ObjectGrid Version 6.1 herstellen. Verwenden Sie Clients der ObjectGrid Version 6.1 für Clients der J2SE Version 1.4.2 oder höher. Der einzige Grund für die Verwendung eines Clients der ObjectGrid Version 6.0.2 ist die Unterstützung von J2SE Version 1.3.

WebSphere eXtreme Scale wurde mit einem Teil der verfügbaren virtuellen Maschinen getestet, aber die Liste der unterstützten JVMs ist nicht exklusiv. Sie können WebSphere eXtreme Scale in jeder Version 1.4.2 oder höher ausführen, aber wenn ein Problem in der JVM auftritt, müssen Sie sich für Unterstützung an den jeweiligen JVM-Anbieter wenden. Verwenden Sie, sofern möglich, die JVM aus der WebSphere-Laufzeitumgebung auf jeder Plattform, die von WebSphere Application Server unterstützt wird.

Java Platform, Standard Edition 6 ist die beste JVM. Die Leistung von Java 2 Platform, Standard Edition, Version 1.4 ist insbesondere in Szenarien, in denen der Collector "gencon" den Ausschlag gibt, sehr schwach. Java Platform Standard Edition 5 bietet eine gute Leistung, aber Java Platform, Standard Edition 6 eine bessere.

Optimierung der Datei "orb.properties"

Es wird empfohlen, die folgende Datei "orb.properties" für Produktionsumgebungen zu verwenden. Diese Datei wurde unter Laborbedingungen in Grids mit bis zu 1500 JVMs verwendet. Die Datei "orb.properties" befindet sich im Ordner "lib" der verwendeten JRE.

```
# Eigenschaften des IBM JDK für den ORB
org.omg.CORBA.ORBClass=com.ibm.CORBA.iiop.ORB
org.omg.CORBA.ORBSingletonClass=com.ibm.rmi.corba.ORBSingleton

# WS-Interceptor
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.objectgrid.corba.ObjectGridInitializer

# Eigenschaften des WS-ORB und der Plug-ins
com.ibm.CORBA.ForceTunnel=never
com.ibm.CORBA.RequestTimeout=10
com.ibm.CORBA.ConnectTimeout=10

# Erforderlich, wenn sehr viele JVMs gleichzeitig eine Verbindung zum Katalog herstellen
com.ibm.CORBA.ServerSocketQueueDepth=2048
```

```
# Clients und Katalogserver können offene Sockets zu allen JVMs haben
com.ibm.CORBA.MaxOpenConnections=1016

# Thread-Pool für die Verarbeitung eingehender Anforderungen, hier 200 Threads
com.ibm.CORBA.ThreadPool.IsGrowable=false
com.ibm.CORBA.ThreadPool.MaximumSize=200
com.ibm.CORBA.ThreadPool.MinimumSize=200
com.ibm.CORBA.ThreadPool.InactivityTimeout=180000

# Keine Aufteilung großer Anforderungen/Antworten in kleinere Blöcke
com.ibm.CORBA.FragmentSize=0
```

Thread-Anzahl

Die Thread-Anzahl ist von verschiedenen Faktoren abhängig. Die Anzahl der Threads, die von einem einzelnen Shard verwaltet werden können, ist begrenzt. Je mehr Shards jede JVM enthält, desto mehr Threads und mehr gemeinsame Zugriffe sind möglich. Jedes zusätzliche Shard liefert weitere nebenläufige Pfade für die Daten. Jedes Shard ist so nebenläufig wie möglich, aber nichtsdestotrotz gibt es einen Grenzwert.

Failover-Erkennung konfigurieren

Sie können das Intervall, in dem das System nach ausgefallenen Servern sucht, mit der Einstellung für das Intervall der Überwachungssignale konfigurieren.

Informationen zu diesem Vorgang

Die Konfiguration des Failovers variiert je nach Typ der verwendeten Umgebung. Wenn Sie eine eigenständige Umgebung verwenden, können Sie das Failover über die Befehlszeile konfigurieren. Wenn Sie eine Umgebung mit WebSphere Application Server Network Deployment verwenden, müssen Sie das Failover über die Administrationskonsole von WebSphere Application Server Network Deployment konfigurieren.

Vorgehensweise

- Failover für eigenständige Umgebungen konfigurieren

  Sie können das Intervall der Überwachungssignale über die Befehlszeile mit dem Parameter **-heartbeat** in der Scriptdatei startOgServer.bat bzw. startOgServer.sh konfigurieren. Setzen Sie den Parameter auf einen der folgenden Werte:

  Tabelle 28. Intervall der Überwachungssignale

Wert	Aktion	Beschreibung
0	Typisch (Standard-einstellung)	Failover werden gewöhnlich innerhalb von 30 Sekunden erkannt.
-1	Aggressiv	Failover werden gewöhnlich innerhalb von 5 Sekunden erkannt.
1	Gelockert	Failover werden gewöhnlich innerhalb von 180 Sekunden erkannt.

  Ein aggressives Intervall der Überwachungssignale kann hilfreich sein, wenn die Prozesse und das Netz stabil sind. Wenn das Netz oder die Prozesse nicht optimal konfiguriert sind, können Überwachungssignale verpasst werden, was zu einer falschen Fehlererkennung führen kann.

- Failover für Umgebungen mit WebSphere Application Server konfigurieren

  Sie können WebSphere Application Server Network Deployment Version 6.0.2 und höher so konfigurieren, dass ein schnelles Failover von WebSphere eXtreme Scale unterstützt wird. Die Standard-Failover-Zeit für permanente Fehler sind

200 Sekunden. Ein permanenter Fehler ist ein physischer Computer- oder Serverabsturz, das Ziehen des Netzkabels oder ein Betriebssystemfehler. Bei Fehlern aufgrund von Prozessabstürzen oder temporären Fehlern findet das Failover gewöhnlich in weniger als einer Sekunde statt. Die Fehlererkennung für temporäre Fehler findet statt, wenn die Netz-Sockets des inaktiven Prozesses für den Server, in dem der Prozess ausgeführt wird, automatisch vom Betriebssystem geschlossen werden.

Überwachungssignalkonfiguration für Stammgruppen

Wenn WebSphere eXtreme Scale in einem Prozess von WebSphere Application Server ausgeführt wird, werden die Failover-Merkmale aus den Stammgruppeneinstellungen des Anwendungsservers übernommen. In den folgenden Abschnitten wird beschrieben, wie Sie die Überwachungssignaleinstellungen der Stammgruppe für verschiedene Versionen von WebSphere Application Server Network Deployment konfigurieren:

– **Stammgruppeneinstellungen für WebSphere Application Server Network Deployment Version 6.x und 7.x aktualisieren**

  Geben Sie das Intervall der Überwachungssignale in WebSphere Application Server Version 6.0 bis Version 6.1.0.12 in Sekunden und ab Version 6.1.0.13 in Millisekunden an. Außerdem müssen Sie die Anzahl verpasster Überwachungssignale angeben. Dieser Wert gibt an, wie viele Überwachungssignale verpasst werden können, bevor eine Peer-JVM als ausgefallen betrachtet wird. Die Erkennungszeit für permanente Fehler entspricht in etwa dem Produkt aus Intervall der Überwachungssignale und Anzahl verpasster Überwachungssignale.

  Diese Eigenschaften werden mit Hilfe von angepassten Eigenschaften in der Stammgruppe über die WebSphere-Administrationskonsole angegeben. Einzelheiten zur Konfiguration finden Sie im Abschnitt Angepasste Eigenschaften der Stammgruppe. Diese Eigenschaften müssen für alle Stammgruppen angegeben werden, die von Anwendungen verwendet werden:

  - Das Intervall der Überwachungssignale wird mit der angepassten Eigenschaft IBM_CS_FD_PERIOD_SEC (in Sekunden) bzw. der angepassten Eigenschaft IBM_CS_FD_PERIOD_MILLIS (in Millisekunden) (erfordert Version 6.1.0.13 oder höher) angegeben.
  - Die Anzahl verpasster Überwachungssignale wird mit der angepassten Eigenschaft IBM_CS_FD_CONSECUTIVE_MISSED angegeben.

  Der Standardwert für die Eigenschaft IBM_CS_FD_PERIOD_SEC ist 20, und der Standardwert für die Eigenschaft IBM_CS_FD_CONSECUTIVE_MISSED ist 10. Wenn Sie die Eigenschaft IBM_CS_FD_PERIOD_MILLIS angeben, überschreibt diese jede definierte angepasste Eigenschaft IBM_CS_FD_PERIOD_SEC. Die Werte dieser Eigenschaften sind positive ganze Zahlen.

  Verwenden Sie die folgenden Einstellungen, um eine Erkennungszeit von 1500 ms für Server der WebSphere Application Server Network Deployment Version 6.x zu erzielen:

  - Setzen Sie `IBM_CS_FD_PERIOD_MILLIS = 750` (WebSphere Application Server Network Deployment Version 6.1.0.13 und höher).
  - Setzen Sie `IBM_CS_FD_CONSECUTIVE_MISSED = 2`.

– **Stammgruppeneinstellungen für WebSphere Application Server Network Deployment Version 7.0 aktualisieren**

  WebSphere Application Server Network Deployment Version 7.0 stellt zwei Stammgruppeneinstellungen bereit, die angepasst werden können, um die Failover-Erkennungszeit zu erhöhen oder zu verringern:

- **Übertragungsintervall für Überwachungssignale.** Der Standardwert sind 30.000 Millisekunden.
- **Überwachungssignalzeitlimit.** Der Standardwert sind 180.000 Millisekunden.

Weitere Einzelheiten zum Ändern dieser Einstellungen finden Sie im Information Center von WebSphere Application Server Network Deployment unter "Einstellungen für die Erkennung und Fehlererkennung".

Verwenden Sie die folgenden Einstellungen, um eine Fehlererkennungszeit von 1500 ms für Server der WebSphere Application Server Network Deployment Version 7 zu erzielen:

- Setzen Sie das Übertragungsintervall für Überwachungssignale auf 750 Millisekunden.
- Setzen Sie das Überwachungssignalzeitlimit auf 1500 Millisekunden.

Nächste Schritte

Wenn Sie diese Einstellungen ändern, um kürzere Failover-Zeiten anzugeben, müssen verschiedene Probleme bei der Systemoptimierung beachtet werden. Java ist keine Echtzeitumgebung. Es ist möglich, dass Threads verzögert werden, wenn die JVM lange Garbage-Collection-Zeiten verzeichnet. Threads können auch verzögert werden, wenn die Maschine, auf der die JVM ausgeführt wird, unter hoher Last steht (durch die JVM selbst oder durch andere Prozesse, die auf der Maschine ausgeführt werden). Wenn Threads verzögert werden, werden Überwachungssignale möglicherweise nicht rechtzeitig gesendet. Im schlimmsten Fall werden die durch die erforderliche Failover-Zeit verzögert. Wenn Threads verzögert werden, treten falsche Fehlererkennungen auf. Das System muss optimiert und dimensioniert werden, um sicherzustellen, dass falsche Fehlererkennungen in der Produktionsumgebung nicht auftreten. Dies kann am Zuverlässigsten durch angemessene Lasttests sichergestellt werden.

Anmerkung: Die aktuelle Version von eXtreme Scale unterstützt WebSphere Real Time.

Fehlererkennungstypen

WebSphere eXtreme Scale kann zuverlässig Fehler erkennen.

Austausch von Überwachungssignalen

1. Sockets werden zwischen Java Virtual Machines bleiben offen, und wenn ein Socket unerwartet geschlossen wird, wird dieses unerwartete Schließen als Fehler von der Peer-JVM erkannt. Bei dieser Erkennung werden Fehlerfälle wie beispielsweise das sehr schnelle Beenden der Java Virtual Machine abgefangen. Außerdem unterstützt dieser Erkennungstyp die Wiederherstellung nach solchen Fehlern in gewöhnlich weniger als einer Sekunde.
2. Weitere Typen von Fehlern sind Betriebssystempanik, physischer Ausfall eines Servers und Netzfehler. Diese Fehler werden über den Austausch von Überwachungssignalen erkannt.

Überwachungssignale werden in regelmäßigen Abständen von zwei Prozessen ausgetauscht. Wenn eine festgelegte Anzahl an Überwachungssignalen verpasst wird, wird von einem Fehler ausgegangen. Mit diesem Ansatz werden Fehler in N*M Sekunden erkannt, wobei N für die Anzahl verpasster Überwachungssignal und M für das Intervall steht, in dem die Überwachungssignale gesendet werden. Die direkte Festlegung von M und N wird nicht unterstützt. Stattdessen wird ein so genannter Slider-Mechanismus verwendet, der die Verwendung eines Bereichs getes-

teter Kombinationen von M und N ermöglicht.

Fehler

Ein Prozess kann auf verschiedene Arten fehlschlagen. Ein Prozess kann fehlschlagen, weil eine Ressourcengrenze erreicht wurde, wie z. B. die maximale Größe des Heap-Speichers, oder weil eine Prozesssteuerungslogik den Prozess beendet hat. Das Betriebssystem kann ausfallen, was dazu führt, dass alle auf dem System ausgeführten Prozesse verloren gehen. Die Hardware, wie z. B. die Netzschnittstellenkarte, kann (wenn auch weniger häufig) ausfallen, was zur Trennung des Betriebssystems vom Netz führen kann. In diesem Kontext können all diese Fehler in zwei Kategorien eingeteilt werden: Prozessfehler und Konnektivitätsverlust.

Prozessfehler

WebSphere eXtreme Scale reagiert sehr schnell auf Prozessfehler. Wenn ein Prozess fehlschlägt, ist das Betriebssystem für die Bereinigung aller übrig gebliebenen Ressourcen verantwortlich, die vom Prozess verwendet wurden. Diese Bereinigung umfasst die Portzuordnung und die Konnektivität. Wenn ein Prozess fehlschlägt, wird sofort ein Signal über die von diesem Prozess verwendeten Verbindungen gesendet, um jede einzelne Verbindung zu schließen.

Konnektivitätsverlust

Ein Konnektivitätsverlust tritt auf, wenn die Verbindung des Betriebssystems zum Netz getrennt wird. Infolgedessen kann das Betriebssystem keine Signale an andere Prozesse senden. Es gibt mehrere Gründe für einen Konnektivitätsverlust, die jedoch in zwei Kategorien eingeteilt werden können: Hostausfall und Isolierung.

Hostausfall

Wenn die Stromversorgung einer Hostmachine unterbrochen wird, fällt der Host sofort aus.

Isolierung

Dieses Szenario stellt die komplizierteste Fehlerbedingung für Software dar. Die Behebung einer solchen Fehlerbedingung stellt sich so schwierig dar, weil davon ausgegangen wird, dass der Prozess nicht verfügbar ist, obwohl dies gar nicht der Fall ist.

Containerausfall

Containerausfälle werden im Allgemeinen von Peer-Containern über den Stammgruppenmechanismus erkannt. Wenn ein Container oder eine Gruppe von Containern ausfällt, migriert der Katalogservice die Shards aus den betroffenen Containern. Der Katalogservice sucht zuerst nach einem synchronen Replikat, bevor er die Migration auf ein asynchrones Replikat durchführt. Nach der Migration der primären Shards in die neuen Hostcontainer durchsucht der Katalogservice die neuen Hostcontainer nach den Replikaten, die jetzt fehlen.

Anmerkung: Containerisolierung - Der Katalogservice migriert Shards aus Containern, wenn der Container als nicht verfügbar erkannt wird. Wenn diese Container wieder verfügbar werden, berücksichtigt der Katalogservice sie bei der Verteilung wie beim normalen Startablauf.

Latenzzeit für Erkennung von Containerausfällen

Ausfälle können in die folgenden beiden Kategorien eingeteilt werden: temporäre Ausfälle und permanente Ausfälle. Temporäre Ausfälle werden gewöhnlich durch einen Prozessfehler verursacht. Solche Ausfälle werden vom Betriebssystem erkannt, das genutzte Ressourcen, wie z. B. Netz-Sockets, sehr schnell wiederherstellen kann. Gewöhnlich werden temporäre Ausfälle in weniger als einer Sekunde erkannt. Die Erkennung permanenter Ausfälle kann unter Verwendung der Standardoptimierung für Überwachungssignale bis zu 200 Sekunden dauern. Zu solchen Ausfällen gehören physische Ausfälle von Maschinen, das Ziehen von Netzkabeln und Betriebssystemausfälle. Somit muss eXtreme Scale darauf vertrauen, dass permanente Ausfälle durch den Austausch von Überwachungssignalen erkannt werden, der konfiguriert werden kann.

Mehrere Containerausfälle

Ein Replikat wird niemals in demselben Prozess wie das primäre Shard ausgeführt, da dies beim Verlust des Prozesses zu einem Verlust des primären Shards und des Replikats führen würde. Die Implementierungsrichtlinie definiert ein Attribut, das der Katalogservice verwendet, um festzustellen, ob ein Replikat auf derselben Maschine wie das primäre Shard platziert werden kann. In einer Entwicklungsumgebung auf einer einzigen Maschine können Sie zwei Container verwenden und die Daten des einen Containers im jeweils anderen replizieren. In Produktionsumgebungen ist die Verwendung einer einzigen Maschine unzureichend, weil der Verlust dieses Hosts zum Verlust beider Container führt. Zum Wechsel vom Entwicklungsmodus auf einer einzigen Maschine in den Produktionsmodus mit mehreren Maschinen müssen Sie den Entwicklungsmodus in der Konfigurationsdatei der Implementierungsrichtlinie inaktivieren.

Ausfall des Katalogservice

Da das Katalogservice-Grid ein eXtreme-Scale-Grid ist, wird der Stammgruppenmechanismus auch hier auf dieselbe Weise wie beim Containerausfallprozess verwendet. Der Hauptunterschied besteht darin, dass die Katalogservicedomäne einen Peer-Auswahlprozess für die Definition des primären Shards an Stelle des Katalogservicealgorithmus verwendet, der für die Container verwendet wird.

Beachten Sie, dass der Verteilungsservice und der Stammgruppierungsservice Services vom Typ "1 von N" sind, der Positionsservice und die Verwaltung hingegen überall ausgeführt werden. Ein Service vom Typ "1 von N" wird nur in einem einzigen Member der HA-Gruppe ausgeführt. Der Positionsservice und der Verwaltungsservice werden in allen Membern der HA-Gruppe ausgeführt. Der Verteilungsservice und der Stammgruppierungsservice sind Singletons, weil sie für das Layout des Systems verantwortlich sind. Der Positionsservice und die Verwaltung sind Services, die ausschließlich im Lesezugriff arbeiten und zur Unterstützung der Skalierbarkeit überall vorhanden sind.

Der Katalogservice verwendet die Replikation für seine eigene Fehlertoleranz. Wenn ein Katalogserviceprozess fehlschlägt, muss der Service erneut gestartet werden, um das System in einem Zustand wiederherzustellen, der die gewünschte Stufe der Verfügbarkeit bietet. Schlagen alle Prozesse, in denen der Katalogservice ausgeführt wird, fehl, verliert eXtreme Scale kritische Daten. Nach diesem Fehler müssen alle Container erneut gestartet werden. Da der Katalogservice in vielen Prozessen ausgeführt werden kann, ist das Auftreten dieses Fehlers eher unwahrscheinlich. Wenn Sie jedoch alle Prozesse auf einer einzigen Maschine, in einem einzigen Blade-Gehäuse oder über einen einzigen Netz-Switch ausführen, ist die

Wahrscheinlich eines Fehlers hoch. Versuchen Sie, bekannte Fehlermodi auf Maschinen, auf denen der Katalogservice ausgeführt wird, zu beseitigen, um die Fehlerwahrscheinlichkeit zu reduzieren.

Tabelle 29. Zusammenfassung der Fehlererkennung und Wiederherstellung

Verlusttyp	Erkennungsmechanismus	Wiederherstellungsmethode
Prozessverlust	Ein-/Ausgabe	Neustart
Serververlust	Überwachungssignal	Neustart
Netzausfall	Überwachungssignal	Netz und Verbindung wiederherstellen
Serverseitige Blockierung	Überwachungssignal	Server stoppen und erneut starten
Server ausgelastet	Überwachungssignal	Warten, bis Server wieder verfügbar ist

WebSphere Real Time verwenden

Die Verwendung von WebSphere eXtreme Scale mit WebSphere Real Time erhöht die Konsistenz und die Voraussagbarkeit, verringert jedoch den Leistungsdurchsatz im Vergleich mit der Standard-Garbage-Collection-Richtlinie, die in der standardmäßig verwendeten IBM Java™ SE Runtime Environment (JRE) genutzt wird. Die Kosten/Nutzen-Aussage kann variieren. WebSphere eXtreme Scale erstellt zahlreiche temporäre Objekte für jede Transaktion. Die temporären Objekte beziehen sich auf Anforderungen, Antworten, Protokollfolgen und Sitzungen. Wenn Sie WebSphere Real Time nicht verwenden, kann die Antwortzeit auf mehrere Hundert Millisekunden ansteigen. Der Einsatz von WebSphere Real Time mit WebSphere eXtreme Scale kann die Effizienz der Garbage-Collection steigern und die Antwortzeit auf 10 % der Antwortzeit in der eigenständigen Konfiguration verringern.

WebSphere Real Time in einer eigenständigen Umgebung

Sie können WebSphere Real Time mit WebSphere eXtreme Scale verwenden. Durch die Aktivierung von WebSphere Real Time erreichen Sie eine vorhersehbarere Garbage-Collection mit stabilen, konsistenten Antwortzeiten und Transaktionsdurchsätzen in einer eigenständigen eXteme-Scale-Umgebung.

Vorteile von WebSphere Real Time

WebSphere eXtreme Scale erstellt zahlreiche temporäre Objekte für jede Transaktion. Die temporären Objekte beziehen sich auf Anforderungen, Antworten, Protokollfolgen und Sitzungen. Wenn Sie WebSphere Real Time nicht verwenden, kann die Antwortzeit auf mehrere Hundert Millisekunden ansteigen. Der Einsatz von WebSphere Real Time mit WebSphere eXtreme Scale kann die Effizienz der Garbage-Collection steigern und die Antwortzeit auf 10 % der Antwortzeit in der eigenständigen Konfiguration verringern.

WebSphere Real Time aktivieren

Installieren Sie WebSphere Real Time und die eigenständige Konfiguration von WebSphere eXtreme Scale auf den Computern, auf denen Sie eXtreme Scale ausführen möchten. Setzen Sie die Umgebungsvariable JAVA_HOME so, dass sie auf eine Standard-JRE (Java SE Runtime Environment) zeigt.

Setzen Sie die Umgebungsvariable JAVA_HOME so, dass sie auf das installierte Produkt WebSphere Real Time zeigt. Aktivieren Sie WebSphere Real Time anschließend wie folgt.

1. Editieren Sie die Datei `objectgridRoot/bin/setupCmdLine.sh | .bat` der eigenständigen Installation, indem Sie das Kommentarzeichen aus der folgenden Zeile entfernen:

   ```
   WXS_REAL_TIME_JAVA="-Xrealtime -Xgcpolicy:metronome
   -Xgc:targetUtilization=80"
   ```

2. Speichern Sie die Datei.

Jetzt haben Sie WebSphere Real Time aktiviert. Wenn Sie WebSphere Real Time inaktivieren möchten, können Sie der Zeile das Kommentarzeichen wieder hinzufügen.

Bewährte Verfahren

Wenn Sie WebSphere Real Time einsetzen, sind die Antwortzeiten von eXtreme-Scale-Transaktionen vorhersehbarer. Die Ergebnisse zeigen, dass sich die Abweichung der Antwortzeit einer eXtreme-Scale-Transaktion mit WebSphere Real Time im Vergleich zum Standard-Java mit dem Standard-Garbage-Collector erheblich verbessert. Die Aktivierung von WebSphere Real Time mit eXtreme Scale ist optimal, wenn Stabilität und Antwortzeiten Ihrer Anwendung von entscheidender Bedeutung sind.

Die in diesem Abschnitt beschriebenen bewährten Verfahren verdeutlichen, wie WebSphere eXtreme Scale durch Optimierung und Codeverfahren für die erwartete Last effizienter gemacht werden kann.

- Legen Sie die richtige Prozessorbelegungsstufe für Ihre Anwendung und den Garbage-Collector fest.

  WebSphere Real Time bietet die Möglichkeit, die Prozessorbelegung zu steuern, so dass die Auswirkungen der Garbage-Collection auf Ihre Anwendung kontrolliert und minimiert werden. Verwenden Sie den Parameter `-Xgc:targetUtilization=NN`, um `NN` Prozent der Prozessorkapazität festzulegen, die alle 20 Sekunden von Ihrer Anwendung belegt werden. Der Standardwert für WebSphere eXtreme Scale ist 80 %, aber Sie können das Script in der Datei `objectgridRoot/bin/setupCmdLine.sh` ändern und eine andere Zahl festlegen, wie z. B. `70`, womit Sie dem Garbage-Collector mehr Prozessorkapazität zuweisen. Implementieren Sie genügend Server, um die Prozessorbelegung für Ihre Anwendungen unter 80 % zu halten.
- Legen Sie einen höheren Wert für die Heap-Speichergröße fest.

  WebSphere Real Time belegt mehr Hauptspeicher als reguläres Java. Planen Sie WebSphere eXtreme Scale deshalb mit einem größeren Heap-Speicher, und legen Sie die Heap-Speichergröße beim Start der Katalogserver und Container mit dem Parameter `–jvmArgs –XmxNNNM` im Befehl ogStartServer fest. Sie können den Parameter `–jvmArgs –Xmx500M` beispielsweise verwenden, um Katalogserver zu starten und eine entsprechende Hauptspeichergröße zum Starten der Container zu verwenden. Sie können die Hauptspeichergröße auf 60-70 % der erwarteten Datenmenge pro JVM setzen. Wenn Sie diesen Wert nicht festlegen, kann ein Fehler des Typs "OutOfMemoryError" auftreten. Optional können Sie auch den Parameter `–jvmArgs –Xgc:noSynchronousGCOnOOM` verwenden, um ein nicht deterministisches Verhalten zu verhindern, wenn in der JVM eine abnormale Speicherbedingung auftritt.
- Threads für die Garbage-Collection anpassen.

WebSphere eXtreme Scale erstellt eine Vielzahl temporärer Objekte für jede Transaktion und jeden RPC-Thread (Remote Procedure Call). Die Garbage-Collection bietet eine bessere bessere Leistung, wenn Ihr Computer genügtend Prozessorzyklen besitzt. Die Standardanzahl der Threads ist 1. Sie können die Anzahl der Threads mit dem Argument `–Xgcthreads n` anpassen. Der vorgeschlagene Wert für dieses Argument ist die Anzahl der verfügbaren Kerne unter Berücksichtigung der Anzahl der Java Virtual Machines pro Computer.

- Leistung für Anwendungen mit kurzer Laufzeit mit WebSphere eXtreme Scale anpassen.

  WebSphere Real Time ist für Anwendungen mit langer Laufzeit optimiert. Gewöhnlich müssen Transaktionen von WebSphere eXtreme Scale kontinuierlich über einen Zeitraum von two Stunden hinweg ausgeführt werden, um zuverlässige Leistungsdaten zu erhalten. Sie können den Parameter `–Xquickstart` verwenden, um die Leistung Ihrer Anwendungen mit kurzer Laufzeit zu verbessern. Dieser Parameter weist den JIT-Compiler an, die untere Stufe der Optimierung zu verwenden.

- Clientwarteschlange von WebSphere eXtreme Scale und Clientvermittlung von WebSphere eXtreme Scale minimieren

  Der Hauptvorteil der Verwendung von WebSphere eXtreme Scale mit WebSphere Real Time ist eine hoch zuverlässige Transaktionsantwortzeit, die gewöhnlich erhebliche Verbesserungen bei der Abweichung der Transaktionsantwortzeiten zur Folge hat. Alle in die Warteschlange eingereihten Clientanforderungen und Clientanforderungsvermittlungen über andere Software wirken sich auf die Antwortzeit aus, die außerhalb der Kontrolle von WebSphere Real Time und WebSphere eXtreme Scale liegt. Sie sollten müssen Ihre Thread- und Socket-Parameter ändern, um eine stabile und gleichmäßige Last ohne größere Verzögerungen zu erzielen und die Länge der Warteschlangen zu verringern.

- Anwendungen von WebSphere eXtreme Scale so schreiben, dass sie das Threading von WebSphere Real Time verwenden

  Sie können ohne Änderung Ihrer Anwendung hoch zuverlässige Transaktionsantwortzeiten in WebSphere eXtreme Scale mit erheblichen Verbesserungen bei der Antwortzeitabweichung erreichen. Außerdem können Sie Threading für Ihre transaktionsorientierten Anwendungen (von regulären Java-Threads zu Realtime-Threads) nutzen, das Ihnen eine bessere Steuerung der Thread-Prioritäten und der Planung bietet.

  Ihre Anwendung enthält derzeit den folgenden Code:

  ```
  public class WXSCacheAppImpl extends Thread implements WXSCacheAppIF
  ```

  Optional können Sie diesen Code durch Folgenden ersetzen.

  ```
  public class WXSCacheAppImpl extends RealtimeThread implements
  WXSCacheAppIF
  ```

WebSphere Real Time in WebSphere Application Server

Sie können WebSphere® Real Time mit eXtreme Scale in einer Umgebung von WebSphere Application Server Network Deployment Version 7.0 verwenden. Durch die Aktivierung von WebSphere Real Time erreichen Sie eine vorhersehbarere Garbage-Collection mit stabilen, konsistenten Antwortzeiten und Transaktionsdurchsätzen.

Vorteile

Die Verwendung von WebSphere eXtreme Scale mit WebSphere Real Time erhöht die Konsistenz und die Vorhersagbarkeit, verringert jedoch den Leistungsdurchsatz im Vergleich mit der Standard-Garbage-Collection-Richtlinie, die in der standard-

mäßig verwendeten IBM Java™ SE Runtime Environment (JRE) genutzt wird. Die Kosten/Nutzen-Aussage kann je nach Kriterien variieren. Im Folgenden sind einige der Hauptkriterien aufgeführt:

- Serverkapazitäten - Verfügbarer Hauptspeicher, CPU-Geschwindigkeit und -kapazität, Netzgeschwindigkeit und -belegung
- Serverlast – CPU-Dauerlast, CPU-Spitzenlast
- Java-Konfiguration – Größe des Heap-Speichers, Zielbelegung, Garbage-Collection-Threads
- Kopiermoduskonfiguration von WebSphere eXtreme Scale – Bytefeldgruppe vs. POJO-Speicher
- Anwendungsspezifikationen – Thread-Belegung, Antwortzeitvoraussetzungen und -toleranz, Objektgröße usw.

Zusätzlich zu der in WebSphere Real Time verfügbaren Metronom-Garbage-Collection-Richtlinie gibt es optionale Garbage-Collection-Richtlinien in der Standard-JRE von IBM (Java™ SE Runtime Environment). Diese Richtlinien, optthruput (Standard), gencon, optavgpause und subpool, sind speziell für die verschiedenartigen Anwendungsanforderungen und -umgebungen konzipiert. Weitere Informationen zu diesen Richtlinien finden Sie unter „JVM-Optimierung für WebSphere eXtreme Scale" auf Seite 435. Je nach Anwendungs- und Umgebungsanforderungen, Ressourcen und Einschränkungen, können Sie durch die Prototyperstellung einer oder mehrerer dieser Garbage-Collection-Richtlinien sicherstellen, dass die Anforderungen erfüllt werden, und eine optimale Richtlinie festlegen.

Möglichkeiten mit WebSphere Application Server Network Deployment

1. Im Folgenden sind einige der unterstützten Versionen aufgelistet:
   - WebSphere Application Server Network Deployment Version 7.0.0.5 und höher
   - WebSphere Real Time V2 SR2 for Linux und höher. Weitere Informationen finden Sie auf der Webseite von IBM WebSphere Real Time V2 for Linux.
   - WebSphere eXtreme Scale Version 7.0.0.0 und höher
   - 32- und 64-Bit-Linux-Betriebssysteme
2. WebSphere-eXtreme-Scale-Server können nicht mit einem Deployment Manager von WebSphere Application Server kollokiert werden.
3. Real Time unterstützt keine Deployment Manager.
4. Real Time unterstützt keine WebSphere-Node-Agents.

WebSphere Real Time aktivieren

Installieren Sie WebSphere Real Time und WebSphere eXtreme Scale auf den Computern, auf denen Sie eXtreme Scale ausführen möchten. Aktualisieren Sie WebSphere Real Time Java auf SR2.

Sie können die JVM-Einstellungen für jeden Server über die Konsole von WebSphere Application Server Version 7.0 wie folgt festlegen.

Wählen Sie **Server → Servertypen → WebSphere-Anwendungsserver → <erforderlicher installierter Server>** aus.

Wählen Sie auf der daraufhin angezeigten Seite "Prozessdefinition" aus.

Klicken Sie auf der nächsten Seite oben in der rechten Spalte auf Java Virtual Machine. (Hier können Sie die Größe des Heap-Speicher, die Garbage-Collection und weitere Flags für jeden Server definieren.)

Setzen Sie die folgenden Flags im Feld "Generische JVM-Argumente":

```
-Xrealtime -Xgcpolicy:metronome  -Xnocompressedrefs -Xgc:targetUtilization=80
```

Wenden Sie die Änderungen an, und speichern Sie sie.

Wenn Sie Real Time in WebSphere Application Server 7.0 mit eXtreme-Scale-Servern, einschließlich der zuvor genannten JVM-Flags, verwenden möchten, müssen Sie eine Umgebungsvariable JAVA_HOME erstellen.

Setzen Sie JAVA_HOME wie folgt.

1. Klicken Sie auf "Umgebung".
2. Wählen Sie "WebSphere-Variablen" aus.
3. Stellen Sie sicher, dass das Feld "Alle Geltungsbereiche" unter "Geltungsbereich anzeigen" ausgewählt ist.
4. Wählen Sie den erforderlichen Server in der Dropdown-Liste aus. (Wählen Sie weder Deployment-Manager- noch Node-Agent-Server aus.)
5. Wenn die Umgebungsvariable JAVA_HOME nicht aufgelistet ist, wählen Sie "Neu" aus, und geben Sie JAVA_HOME als Variablennamen an. Geben Sie im Feld "Wert" den vollständig qualifizierten Pfadnamen von Real Time an.
6. Wenden Sie Ihre Änderung an, und speichern Sie sie.

Bewährte Verfahren

Ein Satz bewährter Verfahren ist im Abschnitt "Bewährte Verfahren" unter „WebSphere Real Time verwenden” auf Seite 444 beschrieben. In dieser Liste bewährter Verfahren sind verschiedene wichtige Änderungen für eine eigenständige Umgebung von WebSphere eXtreme Scale zu beachten, wenn das Produkt in einer Umgebung von WebSphere Application Server Network Deployment installiert wird.

Sie müssen alle zusätzlichen JVM-Befehlszeilenparameter an derselben Position wie die im vorherigen Abschnitt beschriebenen Parameter für die Garbage-Collection-Richtlinie platzieren.

Ein annehmbarer Anfangszielwert für die Dauerlast der Prozessoren ist 50 % mit kurzfristigen Spitzenlasten bis zu 75 %. Wenn die Last diese Werte überschreitet, müssen Sie zusätzliche Kapazitäten hinzufügen, bevor Sie feststellen, dass Vorhersagbarkeit und Konsistenz messbar nachlassen. Sie können die Leistung geringfügig erhöhen, wenn Sie längere Antwortzeiten tolerieren. Das Überschreiten eines Schwellenwerts von 80 % führt häufig zu einer signifikanten Verschlechterung von Konsistenz und Vorhersagbarkeit.

Dynamischen Cacheprovider optimieren

Der dynamische Cacheprovider von WebSphere eXtreme Scale unterstützt die folgenden Konfigurationsparameter für die Leistungsoptimierung.

Informationen zu diesem Vorgang

- **com.ibm.websphere.xs.dynacache.ignore_value_in_change_event**: Wenn Sie einen Listener für Änderungsereignisse beim dynamischen Cacheprovider registrieren und eine ChangeEvent-Instanz generieren, entstehen Kosten für die Entse-

rialisierung des Cacheeintrags, so dass der Wert in das ChangeEvent gestellt werden kann. Die Entserialisierung des Cacheeintrags wird beim Generieren von CacheEvents übersprungen, wenn Sie diesen optionalen Parameter in der Cacheinstanz auf `true` setzen. Der zurückgegebene Wert ist null, wenn eine Entfernungsoperation durchgeführt wird, bzw. eine Bytefeldgruppe, die die serialisierte Form des Objekts enthält. InvalidationEvent-Instanzen bringen ähnliche Leistungseinbußen mit sich, die Sie verhindern können, indem Sie "com.ibm.ws.cache.CacheConfig.ignoreValueInInvalidationEvent" auf `true` setzen.

- **com.ibm.websphere.xs.dynacache.enable_compression**: Standardmäßig komprimiert der dynamische Cacheprovider von eXtreme Scale die Cacheeinträge im Hauptspeicher, um die Cachedichte zu erhöhen. Diese Komprimierung kann den Speicherbedarf für Anwendung wie Servlet-Caching erheblich verringern. Wenn Sie wissen, dass die meisten Cachedaten nicht komprimierbar sind, sollten Sie diese Einstellung auf "false" setzen.

Agent für die Messung der Cachegröße im Hinblick auf genaue Schätzungen der Speicherbelegung optimieren

In Version 7.1 unterstützt WebSphere eXtreme Scale erstmalig die Messung der Speicherbelegung durch BackingMaps in verteilten Grids. (Für lokale Grid-Instanzen wird die Messung der Speicherbelegung nicht unterstützt.) In den meisten Fällen liegt der von WebSphere eXtreme Scale für eine bestimmte Map berichtete Wert sehr nahe bei dem Wert, der von der Heap-Speicherauszugsanalyse berichtet wird. Die Komplexität eines Map-Objekts kann einer genauen Messung im Wege stehen. Im Protokoll wird die Nachricht CWOBJ4543 für jedes Cacheeintragsobjekt angezeigt, das nicht genau gemessen werden kann, weil es zu komplex ist. Die Einhaltung der im Folgenden beschriebenen bewährten Verfahren zur Vermeidung einer unnötigen Map-Komplexität kann die Genauigkeit der Cachemesswerte erhöhen.

Vorgehensweise

- Aktivieren Sie den Agenten für die Messung.

  Wenn Sie eine JVM der Java Version 5 oder höher verwenden, können Sie den Agenten für die Messung verwenden, der WebSphere eXtreme Scale ermöglicht, zusätzliche Informationen von der JVM anzufordern, um die Schätzungen zu verbessern. Der Agent kann durch Hinzufügen des folgenden Arguments zur JVM-Befehlszeile geladen werden:

  ```
  -javaagent:WXS-Bibliotheksverzeichnis/wxssizeagent.jar
  ```

  Für eine integrierte Topologie fügen Sie das Argument der Befehlszeile für den Prozess von WebSphere Application Server hinzu.

  Für eine verteilte Topologie fügen Sie das Argument der Befehlszeile der Prozesse von eXtreme Scale (Container) und des Prozesses von WebSphere Application Server hinzu.

  Wenn der Agent ordnungsgemäß geladen wird, erscheint die folgende Nachricht in der Datei `SystemOut.log`.

  ```
  CWOBJ4541I: Die erweiterte Speichergrößenanpassung für die BackingMap ist aktiviert.
  ```

- Verwenden Sie vorzugsweise und sofern möglich Java-Datentypen anstelle angepasster Datentypen.

  WebSphere eXtreme Scale kann die Speicherkosten der folgenden Typen genau messen:

  – java.lang.String und Arrays, wenn String die Komponentenklasse ist (String[])
  – alle primitiven Wrapper-Typen (Byte, Short, Character, Boolean, Long, Double, Float, Integer) und Arrays, wenn primitive Wrapper der Komponententyp sind (z. B. Integer[], Character[])

- java.math.BigDecimal and java.math.BigInteger und Arrays, wenn diese beiden Klassen der Komponententyp sind (BigInteger[] und BigDecimal[])
- Zeittypen (java.util.Date, java.sql.Date, java.util.Time, java.sql.Timestamp)
- java.util.Calendar und java.util.GregorianCalendar

- Vermeiden Sie Objekt-Internment.

  Wenn ein Objekt in eine Map eingefügt wird, geht WebSphere eXtreme Scale davon aus, dass diese Map die einzige Referenz auf das Objekt und alle Objekte, die von diesem Objekt referenziert werden, enthält. Wenn Sie 1000 angepasste Objekte in eine Map einfügen und jedes eine Referenz auf dieselbe String-Instanz enthält, misst WebSphere eXtreme Scale diese String-Instanz 1000 Mal und überschätzt damit die tatsächliche Größe der Map im Heap-Speicher. WebSphere eXtreme Scale kompensiert dies in den folgenden gängigen Internment-Szenarien ordnungsgemäß:
  - Referenzen auf Java-5-Aufzählungen (enums)
  - Referenzen auf Klassen, die dem typsicheren enum-Muster folgen. Klassen, die diesem Muster folgen, haben nur private Konstrukturen, mindestens ein privates Feld "static final" des eigenen Typs, und wenn sie "Serializable" implementieren, implementieren sie "readResolve()".
  - Internment primitiver Java-5-Wrapper, z. B. Verwendung von Integer.valueOf(1) anstelle von new Integer(1)

  Wenn Sie mit Internment arbeiten müssen, verwenden Sie deshalb eine der vorherigen Techniken.
- Verwenden Sie angepasste Typen mit Bedacht.

  Wenn Sie angepasste Typen verwenden, sollten Sie primitive Datentypen Objekttypen für Felder vorziehen.

  Ziehen Sie außerdem die unter 2 aufgelisteten Objekttypen eigenen angepassten Implementieren vor.

  Beschränken Sie die Objektstruktur bei der Verwendung angepasster Typen auf eine einzige Ebene. Wenn Sie ein angepasstes Objekt in eine Map einfügen, berechnet WebSphere eXtreme Scale nur die Kosten für das eingefügte Objekt, einschließlich aller primitiven Felder und aller direkt im Objekt referenzierten Objekte. WebSphere eXtreme Scale verfolgt keine Referenzen auf untere Ebenen der Objektstruktur. Wenn Sie ein Objekt in die Map einfügen und WebSphere eXtreme Scale Referenzen erkennt, die während des Messprozesses nicht verfolgt wurden, wird eine Nachricht CWOBJ4543 ausgegeben, die den Namen der Klasse enthält, die nicht vollständig gemessen werden konnte. In diesem Fall behandeln Sie die Größenstatistiken in der Map als Trenddaten und nicht als exakte Summe.
- Verwenden Sie, sofern möglich, CopyMode.COPY_TO_BYTES.

  Die Verwendung von CopyMode.COPY_TO_BYTES schaltet alle Unsicherheiten bei der Messung von Werteobjekten aus, die in die Map eingefügt werden, selbst wenn eine Objektstruktur zu viele Ebenen für eine normale Messung hat (was zur Ausgabe der Nachricht CWOBJ4543 führt).

Messung der Cachebelegung

In Release 7.1 kann WebSphere eXtreme Scale erstmalig die Belegung des Java-Heap-Speichers für eine bestimmte BackingMap exakt in Bytes schätzen. Mit dieser Funktion können Sie die Einstellungen für den Heap-Speicher und die Bereinigungsrichtlinien für Ihre Java Virtual Machine korrekt messen. Das Verhalten dieses Features variiert mit der Komplexität der Objekte, die in die BackingMap ein-

gefügt werden, und der Konfiguration der Map. Derzeit wird dieses Feature nur für verteilte Grids unterstützt. Lokale Grid-Instanzen unterstützen die Messung belegter Bytes nicht.

eXtreme Scale speichert alle Daten im Heap-Speicher der JVM-Prozesse, aus denen sich ein Grid zusammensetzt. Der belegte Heap-Speicher für eine bestimmte Map kann in die folgenden Komponenten unterteilt werden:

- Größe aller derzeit in der Map befindlichen Schlüsselobjekte
- Größe aller derzeit in der Map befindlichen Werteobjekte
- Größe aller EvictorData-Objekte, die von Evictor-Plug-ins in dieser Map verwendet werden
- Gemeinkosten für die zugrunde liegende Datenstruktur

Die Anzahl der in den Messstatistiken berichteten belegten Bytes ist die Summe dieser vier Kostenpunkte. Diese Werte werden auf Eintragsbasis in den Map-Operationen "insert" (Einfügen), "update" (Aktualisiere) und "remove" (Entfernen) berechnet, d. h., dass eXtreme Scale immer den aktuellen Wert für die Anzahl der Bytes hat, die eine bestimmte BackingMap belegt.

Wenn Grids partitioniert sind, enthält jede Partition einen Teil der BackingMap. Da die Messstatistiken auf der untersten Ebene des eXtreme-Scale-Codes berechnet werden, verfolgt jede Partition einer BackingMap ihre eigene Größe. Sie können die eXtreme Scale Statistik-APIs verwenden, um die kumulative Größe der Map sowie die Größe der einzelnen Map-Partitionen zu verfolgen.

Im Allgemeinen sollten Sie die Messdaten als Trenddaten und nicht als genaue Messdaten für den von der Map belegten Heap-Speicher behandeln. Wenn sich beispielsweise die berichtete Größe für eine Map von 5 MB auf 10 MB verdoppelt, können Sie davon ausgehen, dass sich die Speicherbelegung der Map verdoppelt hat. Der Messwert von 10 MB kann aus verschiedenen Gründen, die hier beschrieben werden, ungenau sein. Wenn Sie diese Gründe berücksichtigen und den bewährten Verfahren folgen, entspricht die Genauigkeit der Größenmesswerte nahezu der der Nachbearbeitung eines Java-Heap-Speicherauszugs.

Das Hauptproblem mit der Genauigkeit ist darin gegründet, dass das Java-Speichermodell für garantiert genaue Speichermesswerte nicht nicht restriktiv genug ist. Das grundlegende Problem besteht darin, dass ein Objekt aufgrund mehrerer Referenzen im Heap-Speicher verbleiben kann. Wird dieselbe 5-KB-Objektinstanz beispielsweise in drei verschiedene Maps eingefügt, verhindert jede dieser drei Maps die Erfassung des Objekts durch die Garbage-Collection. In dieser Situation ist jeder der folgenden Messwerte vertretbar:

- Die Größe jeder Map erhöht sich um 5 KB.
- Die Größe der ersten Map, in die das Objekt eingefügt wird, erhöht sich um 5 KB.
- Die Größe der anderen beiden Maps erhöht sich nicht. Die Größe jeder Map erhöht sich um einen Anteil der Objektgröße.

Aufgrund dieser Mehrdeutigkeit sollten diese Messwerte als Trenddaten behandelt werden, sofern die Mehrdeutigkeit nicht durch Designoptionen, bewährte Verfahren und Verständnis der Implementierungsoptionen mit diesem Feature ausgeschaltet wird.

eXtreme Scale geht davon aus, dass eine bestimmte Map die einzige Langzeitreferenz auf Schlüssel- und Werteobjekte, die sie enthält, besitzt. Wenn dasselbe 5-KB-

Objekt in drei Maps eingefügt wird, erhöht sich die Größe jeder Map um 5 KB. Die Erhöhung ist gewöhnlich kein Problem, weil das Feature nur für verteilte Grids unterstützt wird. Wenn Sie dasselbe Objekt in drei verschiedene Maps auf einem fernen Client einfügen, erhält jede Map eine eigene Kopie des Objekts. Die Standardtransaktionseinstellungen für den Kopiermodus (COPY MODE) garantieren gewöhnlich, dass jede Map eine eigene Kopie eines bestimmten Objekts erhält.

Das Objekt-Internment ist die größte Herausforderung für die meisten Kundenanwendungen. Objekt-Internment bedeutet, dass der Anwendungscode sicherstellt, dass alle Referenzen auf einen bestimmten Objektwert auf dieselbe Objektinstanz im Heap-Speicher zeigen. Ein Beispiel hierfür ist die folgende Klasse:

```
public class ShippingOrder implements Serializeable,Cloneable{

    public static final STATE_NEW = "new";
    public static final STATE_PROCESSING = "processing";
    public static final STATE_SHIPPED = "shipped";

    private String state;
    private int orderNumber;
private int customerNumber;

public Object clone(){
       ShippingOrder toReturn = new ShippingOrder();
       toReturn.state = this.state;
       toReturn.orderNumber = this.orderNumber;
       toReturn.customerNumber = this.customerNumber;
       return toReturn;
    }

    private void readResolve(){
        if (this.state.equalsIgnoreCase("new")
            this.state = STATE_NEW;
        else if (this.state.equalsIgnoreCase("processing")
            this.state = STATE_PROCESSING;
        else if (this.state.equalsIgnoreCase("shipped")
            this.state = STATE_SHIPPED:
    }
}
```

Unabhängig von der Konfiguration von eXtreme Scale werden die tatsächlichen Kosten für diese Klasse von den Messstatistiken überschätzt. Wenn es eine Million ShippingOrder-Objekte gibt, spiegelt der Messcode die Kosten für eine Million Zeichenfolgen (Strings) wider, die die Statusinformationen enthalten. In Wirklichkeit gibt es nur drei Zeichenfolgen, und dies sind statische Klasseneinträge. Die Speicherkosten für diese Klasseneinträge dürfen den Maps von eXtreme Scale hinzugefügt werden, aber es gibt keine probate Methode, eine solche Situation zur Laufzeit zu erkennen. Es gibt Dutzende von Methoden, mit denen ein ähnliches Internment erreicht werden kann, und deshalb ist die Erkennung solcher Situationen so schwierig. Ein globaler Schutz vor solchen Situationen in eXtreme Scale ist nicht praktikabel. eXtreme Scale kann jedoch einen Schutz für die gängigsten Typen bieten.

eXtreme Scale führt eine automatische Anpassung für Java-5-Aufzählungen (enums) und das typsichere enum-Muster (siehe http://java.sun.com/j2se/1.5.0/docs/guide/language/enums.html).

Zum Optimieren der Speicherbelegung durch Objekt-Internment werden nur interne angepasste Objekte diesen beiden Kategorien zugeordnet. Die Anwendung dieses Verfahrens erhöht die Genauigkeit der Speicherbelegungsstatistiken. Außerdem wurde in Java 5 das automatische Internment für primitive Wrapper-Typen wie In-

teger durch die Verwendung statischer valueOf()-Methoden eingeführt. eXtreme Scale berücksichtigt dieses Internment automatisch.

Verwenden Sie eine der folgenden Methoden, um auf die Speicherbelegungsstatistiken zuzugreifen.

Statistik-API

Verwenden Sie die Methode "MapStatsModule.getUsedBytes()", die Statistiken für eine einzige Map bereitstellt, einschließlich der Anzahl an Einträgen und der Trefferrate.

Einzelheiten finden Sie unter „Statistikmodule” auf Seite 390.

Managed Beans (MBeans)

Verwenden Sie die MBean-Statistik "MapUsedBytes". Sie können verschiedene Typen von JMX-Beans (Java Management Extensions) verwenden, um Implementierungen zu verwalten und zu überwachen. Jede MBean bezieht sich auf eine bestimmte Entität, z. B. eine Map, eXtreme Scale, einen Server, eine Replikationsgruppe oder ein Replikationsgruppen-Member.

Einzelheiten finden Sie unter „Programmgesteuerte Verwaltung mit Managed Beans (MBeans)” auf Seite 360.

PMI-Module (Performance Monitoring Infrastructure)

Sie können die Leistung Ihrer Anwendungen mit den PMI-Modulen überwachen. Verwenden Sie insbesondere die PMI-Module für Container, die in WebSphere Application Server integriert sind.

Einzelheiten finden Sie unter „PMI-Module” auf Seite 398.

Konsole von WebSphere eXtreme Scale

Die in Version 7.1 eingeführte Konsole ermöglicht Ihnen, die Speicherbelegungsstatistiken anzuzeigen. Einzelheiten finden Sie unter „Überwachung mit der Webkonsole” auf Seite 407.

Alle beschriebenen Methoden greifen auf denselben Basismesswert für die Speicherbelegung einer bestimmten BaseMap-Instanz zu. Die Laufzeitumgebung von WebSphere eXtreme Scale versucht (bestmöglich), die von den in der Map gespeicherten Schlüssel- und Werteobjekten belegten Bytes des Heap-Speichers und die Gemeinkosten für die Map selbst zu berechnen. Sie können anzeigen, wie viel Heap-Speicher die einzelnen Maps im gesamten verteilten Grid belegen.

In den meisten Fällen liegt der von WebSphere eXtreme Scale für eine bestimmte Map berichtete Wert sehr nahe bei dem Wert, der von der Heap-Speicherauszugsanalyse berichtet wird. WebSphere eXtreme Scale misst seine eigenen Gemeinkosten genau, kann aber nicht jedes potenzielle Objekt berücksichtigen, dass in eine Map eingefügt wird. Durch die Einhaltung der unter „Agent für die Messung der Cachegröße im Hinblick auf genaue Schätzungen der Speicherbelegung optimieren” auf Seite 449 beschriebenen bewährten Verfahren kann die Genauigkeit der ermittelten Bytemesswerte von WebSphere eXtreme Scale erhöhen.

Kapitel 11. Fehlerbehebung

Zusätzlich zu den in diesem Abschnitt beschriebenen Protokollen, Trace, Nachrichten und Releaseinformationen können Sie Überwachungstools verwenden, um Gegebenheiten zu verstehen, wie z. B. die Position der Daten in der Umgebung, die Verfügbarkeit der Server im Grid usw. Wenn Sie in einer Umgebung mit WebSphere Application Server arbeiten, können Sie Performance Monitoring Infrastructure (PMI) verwenden. Wenn Sie ein einer eigenständigen Umgebung arbeiten, können Sie Überwachungstools anderer Anbieter verwenden, wie z. B. CA Wily Introscope oder Hyperic HQ. Außerdem können Sie das Musterdienstprogramm "xsAdmin" verwenden und anpassen, um Textinformationen zu Ihrer Umgebung anzuzeigen.

Protokolle und Trace

Sie können Protokolle und Trace verwenden, um Ihre Umgebung zu überwachen und Fehler zu beheben. Protokolle werden je nach Konfiguration an unterschiedlichen Positionen gespeichert. Möglicherweise müssen Sie einen Trace für einen Server bereitstellen, wenn Sie mit der IBM Unterstützungsfunktion zusammenarbeiten.

Protokolle mit WebSphere Application Server

Weitere Informationen finden Sie im Information Center von WebSphere Application Server.

Protokolle mit WebSphere eXtreme Scale in einer eigenständigen Umgebung

Bei eigenständigen Katalog- und Containerservern legen Sie die Position der Protokolle und alle Trace-Spezifikationen fest. Die Katalogserverprotokolle werden dort gespeichert, wo Sie den Befehl zum Starten des Servers ausführen.

Protokollposition für Containerserver festlegen

Standardmäßig werden die Protokolle für einen Server in dem Verzeichnis gespeichert, in dem der Befehl zum Starten des Servers ausgeführt wird. Wenn Sie die Server im Verzeichnis *<Ausgangsverzeichnis_von_eXtreme_Scale>*/bin starten, werden die Protokoll- und Trace-Dateien in den Verzeichnissen logs/*<Servername>* des Verzeichnisses bin gespeichert. Wenn Sie eine andere Position für die Protokolle eines Containerservers festlegen möchten, erstellen Sie eine Eigenschaftendatei, z. B. server.properties, mit dem folgenden Inhalt:

```
workingDirectory=<Verzeichnis>
traceSpec=
systemStreamToFileEnabled=true
```

Die Eigenschaft "workingDirectory" ist das Stammverzeichnis für die Protokolle und die optionale Trace-Datei. WebSphere eXtreme Scale erstellt ein Verzeichnis mit dem Namen des Containerservers mit einer Datei SystemOut.log, einer Datei SystemErr.log und einer Trace-Datei, falls die Trace-Erstellung mit der Option "traceSpec" aktiviert wurde. Wenn eine Eigenschaftendatei während des Containerstarts verwendet werden soll, verwenden Sie die Option **-serverProps**, und geben Sie die Position der Servereigenschaftendatei an.

Weitere Informationen finden Sie in den Abschnitten „Eigenständige Server von WebSphere eXtreme Scale starten" auf Seite 335 und „Script "startOgServer"" auf Seite 340.

Häufige Informationsnachrichten, die in der Datei `SystemOut.log` aufgezeichnet werden, sind Bestätigungsnachrichten für den Start. Weitere Informationen zu bestimmten Nachrichten finden Sie im Abschnitt „Nachrichten" auf Seite 460.

Trace-Erstellung mit WebSphere Application Server

Weitere Informationen finden Sie im Information Center von WebSphere Application Server.

Trace-Erstellung für den Katalogservice

Sie können die Trace-Erstellung für einen Katalogservice festlegen, indem Sie während des Katalogservicestarts die Parameter **-traceSpec** und **-traceFile** verwenden. Beispiel:

```
startOgServer.sh catalogServer -traceSpec
ObjectGridPlacement=all=enabled -traceFile
/home/user1/logs/trace.log
```

Wenn Sie den Katalogservice im Verzeichnis *<Ausgangsverzeichnis_von_eXtreme_Scale>*`/bin` starten, werden die Protokolle und Trace-Dateien in einem Verzeichnis `logs/`*<Name_des_Katalogservice>* des Verzeichnisses `bin` gespeichert. Weitere Einzelheiten zum Starten eines Katalogservice finden Sie im Abschnitt „Katalogservice in einer eigenständigen Umgebung starten" auf Seite 335.

Trace für einen eigenständigen Containerserver erstellen

Sie können die Trace-Erstellung für eine Containerserver auf zwei Arten aktivieren. Sie können, wie im Abschnitt zu den Protokollen erläutert, eine Servereigenschaftendatei erstellen, oder Sie können die Trace-Erstellung beim Start über die Befehlszeile aktivieren. Zum Aktivieren der Trace-Erstellung für den Container über eine Servereigenschaftendatei aktualisieren Sie die Eigenschaft **traceSpec** mit der erforderlichen Trace-Spezifikation. Zum Aktivieren der Trace-Erstellung für den Container über Startparameter verwenden Sie die Parameter **-traceSpec** und **-traceFile**. Beispiel:

```
startOgServer.sh c0 -objectGridFile ../xml/myObjectGrid.xml
-deploymentPolicyFile ../xml/myDepPolicy.xml -catalogServiceEndpoints
server1.rchland.ibm.com:2809 -traceSpec
ObjectGridPlacement=all=enabled -traceFile /home/user1/logs/trace.log
```

Wenn Sie den Server im Verzeichnis *<Ausgangsverzeichnis_von_eXtreme_Scale>*`/bin` starten, werden die Protokoll- und Trace-Dateien in den Verzeichnissen `logs/`*<Servername>* des Verzeichnisses `bin` gespeichert. Weitere Einzelheiten zum Starten eines Containerprozesses finden Sie im Abschnitt „Containerprozesse starten" auf Seite 338.

Trace-Erstellung mit der Schnittstelle "ObjectGridManager"

Eine weitere Option ist die Definition der Trace-Erstellung zur Laufzeit über eine ObjectGridManager-Schnittstelle. Die Definition der Trace-Erstellung in einer ObjectGridManager-Schnittstelle kann verwendet werden, um einen Trace für einen eXtreme-Scale-Client zu erstellen, wenn dieser eine Verbindung zu einer eXtreme-Scale-Instanz herstellt und Transaktionen festschreibt. Wenn Sie die Trace-Erstel-

lung in einer ObjectGridManager-Schnittstelle festlegen möchten, geben Sie eine Trace-Spezifikation und ein Trace-Protokoll an.

```
ObjectGridManager manager = ObjectGridManagerFactory.getObjectGridManager();
...
manager.setTraceEnabled(true);
manager.setTraceFileName("logs/myClient.log");
manager.setTraceSpecification("ObjectGridReplication=all=enabled");
```

Trace-Erstellung mit dem Dienstprogramm "xsadmin" aktivieren

Zum Aktivieren der Trace-Erstellung mit dem Dienstprogramm "xsadmin" verwenden Sie die Option **setTraceSpec**. Verwenden Sie das Dienstprogramm "xsadmin", um die Trace-Erstellung für eine eigenständige Umgebung zur Laufzeit und nicht zur Startzeit zu aktivieren. Sie können die Trace-Erstellung für alle Server und Katalogservices aktivieren, oder Sie können die Server nach dem ObjectGrid-Namen usw. filtern. Wenn Sie beispielsweise den ObjectGridReplication-Trace mit Zugriff auf den Katalogserviceserver aktivieren möchten, führen Sie den folgenden Befehl aus:

```
<Ausgangsverzeichnis_von_eXtreme_Scale>/bin>xsadmin.bat -setTraceSpec "ObjectGridReplication=all=enabled"
```

Sie können die Trace-Erstellung auch inaktivieren, indem Sie die Trace-Spezifikation auf `*=all=disabled.` setzen.

Weitere Informationen finden Sie im Abschnitt „Überwachung mit dem Musterdienstprogramm "xsAdmin"" auf Seite 391.

FFDC-Verzeichnis und -Dateien

FFDC-Dateien sind als Debug-Hilfe für die IBM Unterstützungsfunktion bestimmt. Diese Dateien werden möglicherweise von der IBM Unterstützungsfunktion angefordert, wenn ein Problem auftritt.

Diese Dateien befinden sich in einem Verzeichnis mit dem Namen `ffdc`. Das Verzeichnis enthält Dateien wie die folgenden:

```
server2_exception.log
server2_20802080_07.03.05_10.52.18_0.txt
```

Trace-Optionen

Sie können die Trace-Erstellung aktivieren, um der IBM Unterstützungsfunktion Informationen über Ihre Umgebung bereitzustellen.

Informationen zur Trace-Erstellung

Der WebSphere eXtreme Scale-Trace ist in mehrere Komponenten unterteilt. Ähnlich wie bei der Trace-Erstellung in WebSphere Application Server können Sie die zu verwendende Trace-Stufe angeben. Zu den gängigen Trace-Stufen gehören all, debug, entryExit und event.

Im Folgenden sehen Sie ein Beispiel für eine Trace-Zeichenfolge:

```
ObjectGridComponent=level=enabled
```

Sie können Trace-Zeichenfolgen verknüpfen. Verwenden Sie das Symbol `*` (Stern), um einen Platzhalterwert anzugeben, z. B. `ObjectGrid*=all=enabled`. Wenn Sie einen Trace für die IBM Unterstützungsfunktion bereitstellen müssen, ist eine bestimmte Trace-Zeichenfolge erforderlich. So kann beispielsweise die Trace-Zeichenfolge `ObjectGridReplication=debug=enabled` angefordert werden, wenn ein Problem mit der Replikation auftritt.

Trace-Spezifikation

ObjectGrid
Allgemeine Basiscachesteuerkomponente.

ObjectGridCatalogServer
Allgemeiner Katalogservice.

ObjectGridChannel
Statische Kommunikation in der Implementierungstopologie.

7.1+ **ObjectGridClientInfo**
DB2-Clientinformationen.

7.1+ **ObjectGridClientInfoUser**
DB2-Benutzerinformationen.

ObjectgridCORBA
Dynamische Kommunikation in der Implementierungstopologie.

ObjectGridDataGrid
Die API "AgentManager".

ObjectGridDynaCache
Der dynamische Cacheprovider von WebSphere eXtreme Scale.

ObjectGridEntityManager
Die API "EntityManager". Mit der Option "Projector" zu verwenden.

ObjectGridEvictors
Integrierte ObjectGrid-Evictor (Bereinigungsprogramme).

ObjectGridJPA
JPA-Loader (Java Persistence API).

ObjectGridJPACache
JPA-Cache-Plug-ins.

ObjectGridLocking
Sperrenmanager für ObjectGrid-Cacheeinträge.

ObjectGridMBean
Management-Beans.

7.1+ **ObjectGridMonitor**
Infrastruktur für Langzeitüberwachung.

ObjectGridPlacement
Katalogserverservice für Shard-Verteilung.

ObjectGridQuery
ObjectGrid-Abfrage.

ObjectGridReplication
Replikationsservice.

ObjectGridRouting
Details zum Client/Server-Routing.

ObjectGridSecurity
Sicherheits-Trace.

ObjectGridStats
ObjectGrid-Statistiken.

ObjectGridStreamQuery
Die API "Stream Query".

ObjectGridWriteBehind
ObjectGrid-Write-behind.

Projector
Die Steuerkomponente in der API "EntityManager".

QueryEngine
Die Abfragesteuerkomponente für die API "Object Query" und die API "EntityManager Query".

QueryEnginePlan
Diagnose des Abfrageplans.

IBM Support Assistant für WebSphere eXtreme Scale

Sie können IBM Support Assistant verwenden, um Daten zu erfassen, Symptome zu analysieren und auf Produktinformationen zuzugreifen.

IBM Support Assistant Lite

IBM Support Assistant Lite for WebSphere eXtreme Scale unterstützt die automatische Datenerfassung und Symptomanalyse für Problembestimmungsszenarien.

Mit IBM Support Assistant Lite reduziert sich die Zeit, die erforderlich ist, um ein Problem mit den entsprechend definierten Trace-Stufen für Zuverlässigkeit, Verfügbarkeit und Servicefreundlichkeit Trace-Stufen werden automatisch vom Tool gesetzt) zu reproduzieren, um die Fehlerbestimmung zu optimieren. Wenn Sie zusätzliche Unterstützung benötigen, verringert IBM Support Assistant Lite auch den erforderlichen Aufwand für das Senden der entsprechenden Protokollinformationen an die IBM Unterstützungsfunktion.

IBM Support Assistant Lite ist in jeder Installation von WebSphere eXtreme Scale Version 7.1.0 enthalten.

IBM Support Assistant

IBM® Support Assistant (ISA) ermöglicht Ihnen den schnellen Zugriff auf Produkt-, Schulungs- und Unterstützungsressourcen, die Ihnen helfen können, eigenständig Antworten auf Fragen zu finden und Probleme mit IBM Softwareprodukten zu finden, ohne sich an die IBM Unterstützungsfunktion wenden zu müssen. Es werden verschiedene produktspezifische Plug-ins bereitstellt, mit denen Sie IBM Support Assistant für Ihre installierten Produkte anpassen können. IBM Support Assistant kann auch Systemdaten, Protokolldateien und andere Informationen erfassen, die der IBM Unterstützungsfunktion bei der Bestimmung der Ursache eines bestimmten Problems helfen.

IBM Support Assistant ist ein Dienstprogramm, das für die Installation auf der Workstation und nicht für die direkte Installation auf dem System mit dem eXtreme-Scale-Server bestimmt ist. Der Speicher- und Ressourcenbedarf für Assistant kann sich nachteilig auf die Leistung des Systems mit dem eXtreme-Scale-Server auswirken. Die enthaltenen portierbaren Diagnosekomponenten sind so konzipiert, dass sie nur minimale Auswirkungen auf den normalen Betrieb eines Servers haben.

Sie können IBM Support Assistant für folgende Unterstützungszwecke einsetzen:

- Für die Suche von Informationen in Wissens- und Informationsquellen von IBM und anderen Anbietern zu mehreren IBM Produkten, um Antworten auf eine Frage zu finden oder um ein Problem zu lösen.
- Für die Suche zusätzlicher Informationen in produktspezifischen Webressourcen, einschließlich Produkt- und Unterstützungs-Homepages, Kunden-Newsgroups und -Foren, Wissens- und Schulungsressourcen sowie Informationen zur Fehlerbehebung und zu häufig gestellten Fragen.
- Zur Erweiterung Ihrer Möglichkeiten für die Diagnose produktspezifischer Probleme mit den in Support Assistant bereitgestellten zielspezifischen Diagnosetools.
- Für eine vereinfachte Erfassung von Diagnosedaten, die Ihnen und IBM helfen, Probleme zu beheben (Erfassung allgemeiner oder produkt- bzw. symptomspezifischer Daten).
- Zur Unterstützung beim Melden von Problemvorfällen an die IBM Unterstützungsfunktion über eine angepasste Onlineschnittstelle, einschließlich der Möglichkeit, zuvor referenzierte Diagnosedaten oder andere Informationen zu neuen oder vorhandenen Vorfällen anzuhängen.

Und dann können Sie noch das integrierte Updater-Tool verwenden, um Unterstützung für zusätzliche Softwareprodukte und Funktionen zu erhalten, sobald diese verfügbar sind. Zum Einrichten von IBM Support Assistant für WebSphere eXtreme Scale installieren Sie zuerst IBM Support Assistant unter Verwendung der Dateien, die in dem von der Webseite "IBM Support Overview" unter http://www-947.ibm.com/support/entry/portal/Overview/Software/Other_Software/IBM_Support_Assistant heruntergeladenen Image bereitgestellt werden. Anschließend verwenden Sie IBM Support Assistant, um Produktaktualisierungen zu suchen und zu installieren. Sie können auch neue Plug-ins installieren, die für andere IBM Software in Ihrer Umgebung verfügbar sind. Weitere Informationen und die aktuelle Version von IBM Support Assistant sind auf der Webseite von IBM Support Assistant unter http://www.ibm.com/software/support/isa/ verfügbar.

Nachrichten

Wenn Sie in einem Protokoll oder in anderen Teilen der Produktschnittstelle eine Nachricht sehen, können Sie die Nachricht mit Hilfe des Komponentenpräfix suchen, um weitere Informationen zu erhalten.

Nachrichten suchen

Wenn Sie eine Nachricht in einem Protokoll finden, kopieren Sie die Nachrichtennummer mit ihrem Buchstabenpräfix und iher Nummer, und suchen Sie im Information Center danach (z. B. `CWOBJ1526I`). Wenn Sie nach der Nachricht suchen, können Sie eine zusätzliche Erläuterung der Nachricht und Beschreibung möglicher Maßnahmen finden, die Sie zum Beheben des Problems verwenden können.

Einen Index der Produktnachrichten finden Sie im Information Center.

Releaseinformationen

Hier finden Sie Links zur Unterstützungswebsite für das Produkt, zur Produktdokumentation und zum letzten Stand der Updates, Einschränkungen und bekannten Problemen für das Produkt.

- „Zugriff auf den letzten Stand von Updates, Einschränkungen und bekannten Problemen” auf Seite 461

- „Zugriff auf System- und Softwarevoraussetzungen”
- „Zugriff auf die Produktdokumentation ”
- „Zugriff auf die Unterstützungswebsite für das Produkt ”
- „Kontaktaufnahme mit der IBM Softwareunterstützung ”

Zugriff auf den letzten Stand von Updates, Einschränkungen und bekannten Problemen

Die Releaseinformationen sind auf der Produktunterstützungssite als technische Hinweise verfügbar. Eine Liste aller technischen Hinweise für WebSphere eXtreme Scale finden Sie auf der Unterstützungswebseite.Wenn Sie auf die hier bereitgestellten Links klicken, wird die Unterstützungwebseite nach den relevanten Releaseinformationen durchsucht, die dann als Liste zurückgegeben werden.

- 7.1+ Eine Liste der Releaseinformationen für Version 7.1 finden Sie auf der Unterstützungswebseite.
- Eine Liste der Releaseinformationen für Version 7.0 finden Sie auf der Unterstützungswebseite.
- Eine Liste der Releaseinformationen für Version 6.1 finden Sie auf der Wiki-Seite für Releaseinformationen.

Zugriff auf System- und Softwarevoraussetzungen

Die Hardware- und Softwarevoraussetzungen finden Sie auf den folgenden Webseiten:

- Detailed system requirements

Zugriff auf die Produktdokumentation

Das vollständige Informationsset finden Sie auf der Bibliotheksseite.

Zugriff auf die Unterstützungswebsite für das Produkt

Wenn Sie nach den neuesten technischen Hinweisen, Downloads und Korrekturen sowie nach Informationen zur Unterstützung suchen, rufen Sie die Unterstützungswebseite auf.

Kontaktaufnahme mit der IBM Softwareunterstützung

Wenn ein Problem mit dem Produkt auftritt, versuchen Sie zuerst, die folgenden Aktionen auszuführen:

- Führen Sie die in der Produktdokumentation beschriebenen Schritte aus.
- Suchen Sie in der Onlinehilfe nach Referenzliteratur.
- Schlagen Sie die Fehlernachrichten in der Nachrichtenreferenz nach.

Sollten Sie das Problem nicht auf diese Weise lösen können, wenden Sie sich an die technische Unterstützung der IBM.

Bemerkungen

Hinweise auf IBM Produkte, Programme und Services in dieser Veröffentlichung bedeuten nicht, dass IBM diese in allen Ländern, in denen IBM vertreten ist, anbietet. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb der Produkte, Programme oder Fremdservices in Verbindung mit Fremdprodukten und Fremdservices liegt beim Kunden, soweit solche Verbindungen nicht ausdrücklich von IBM bestätigt sind.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense
France

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Marken

Folgende Namen sind Marken der IBM Corporation in den USA und/oder anderen Ländern:

- AIX
- CICS
- Cloudscape
- DB2
- Domino
- IBM
- Lotus
- RACF
- Redbooks
- Tivoli
- WebSphere
- z/OS

Java und alle auf Java basierenden Marken und Logos sind Marken von Sun Microsystems, Inc. in den USA und/oder anderen Ländern.

LINUX ist eine Marke von Linus Torvalds in den USA und/oder anderen Ländern.

Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.

Weitere Unternehmens-, Produkt- oder Servicenamen können Marken anderer Hersteller sein.

Index

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z

Antwort

eXtreme Scale Version 7.1
Administratorhandbuch
WebSphere eXtreme Scale
Administratorhandbuch

Anregungen zur Verbesserung und Ergänzung dieser Veröffentlichung nehmen wir gerne entgegen. Bitte informieren Sie uns über Fehler, ungenaue Darstellungen oder andere Mängel.

Zur Klärung technischer Fragen sowie zu Liefermöglichkeiten und Preisen wenden Sie sich bitte entweder an Ihre IBM Geschäftsstelle, Ihren IBM Geschäftspartner oder Ihren Händler.

Unsere Telefonauskunft "HALLO IBM" (Telefonnr.: 0180 3 313233) steht Ihnen ebenfalls zur Klärung allgemeiner Fragen zur Verfügung.

Kommentare:

Danke für Ihre Bemühungen.

Sie können ihre Kommentare betr. dieser Veröffentlichung wie folgt senden:
- Als Brief an die Postanschrift auf der Rückseite dieses Formulars
- Als E-Mail an die folgende Adresse: ibmterm@de.ibm.com

Name	Adresse
Firma oder Organisation	
Rufnummer	E-Mail-Adresse

Antwort

IBM

IBM Deutschland GmbH
SW TSC Germany

71083 Herrenberg