WebSphere® eXtreme Scale Versión 7.0

Guía de administración

WebSphere® eXtreme Scale Versión 7.0

Guía de administración

Esta edición se aplica a la versión 7, release 0, de WebSphere eXtreme Scale y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones.

Contenido

Figuras

Tablas

Acerca de la *Guía de administración*

El conjunto de documentación de WebSphere eXtreme Scale incluye tres volúmenes que proporcionan la información necesaria para utilizar, programar y administrar el producto WebSphere eXtreme Scale.

Biblioteca de WebSphere eXtreme Scale

La biblioteca de WebSphere eXtreme Scale contiene las siguientes publicaciones:

- La *Guía de administración* contiene la información necesaria para los administradores del sistema, incluido cómo planificar despliegues de aplicaciones, planificar la capacidad, instalar y configurar el producto, iniciar y detener servidores, supervisar el entorno y proteger el entorno.
- La *Guía de programación* contiene información dirigida a los desarrolladores de aplicaciones que indica cómo desarrollar aplicaciones para WebSphere eXtreme Scale utilizando la información de API incluida.
- El *Visión general del producto* contiene una vista de nivel superior de los conceptos de WebSphere eXtreme Scale, incluidos casos de ejemplo y guías de aprendizaje.

Para descargar las publicaciones, vaya a la página de la biblioteca WebSphere eXtreme Scale.

También puede acceder a la misma información de esta biblioteca en el centro de información de WebSphere eXtreme Scale.

Quién debe utilizar esta publicación

Esta publicación está especialmente indicada para administradores del sistema, administradores de seguridad y operadores del sistema.

Estructura de esta publicación

La publicación contiene información sobre los siguientes temas principales:

- **Capítulo 1** incluye información sobre cómo empezar.
- **Capítulo 2** incluye información sobre cómo planificar el despliegue de aplicación.
- **Capítulo 3** incluye información sobre la planificación de la capacidad.
- **Capítulo 4** incluye información sobre cómo instalar el producto.
- **Capítulo 5** incluye información sobre cómo personalizar para la plataforma z/OS.
- **Capítulo 6** incluye información sobre cómo configurar el producto.
- **Capítulo 7** incluye información sobre cómo administrar el entorno.
- **Capítulo 8** incluye información sobre cómo supervisar el entorno de despliegue.
- **Capítulo 9** incluye información sobre cómo proteger el entorno de despliegue.
- **Capítulo 10** incluye información sobre cómo resolver problemas del entorno.
- **Capítulo 11** incluye información del glosario del producto.

Cómo obtener actualizaciones de esta publicación

Puede obtener actualizaciones para esta publicación descargando la versión más reciente desde la página de la biblioteca de WebSphere eXtreme Scale.

Envío de comentarios

Póngase en contacto con el equipo de documentación. ¿Ha encontrado lo que necesita? ¿Ha sido la información precisa y completa? Envíe sus comentarios sobre esta documentación mediante correo electrónico a wasdoc@us.ibm.com.

Capítulo 1. Cómo empezar con WebSphere eXtreme Scale

Tras instalar WebSphere eXtreme Scale en un entorno autónomo, utilice los pasos siguientes como una simple presentación de sus funciones como en una cuadrícula de datos de memoria.

La instalación autónoma de WebSphere eXtreme Scale incluye un ejemplo que puede utilizar para verificar la instalación y para ver cómo se puede utilizar un cliente y una cuadrícula sencilla de eXtreme Scale. El ejemplo de iniciación está en el directorio *installRoot*/ObjectGrid/gettingstarted.

El ejemplo de iniciación proporciona una rápida introducción a las características y al funcionamiento básico de eXtreme Scale. El ejemplo está formado por scripts de shell y de procesos por lotes diseñados para iniciar una cuadrícula sencilla con muy poca necesidad de personalización. Además, un programa cliente, incluido el origen, se proporciona para ejecutar las funciones CRUD (crear, leer, actualizar y suprimir) sencillas en esta cuadrícula básica.

Los scripts y sus funciones

Este ejemplo proporciona los cuatros scripts siguientes:

Los otros scripts llaman al script env.sh|bat para establecer las variables de entorno necesarias. Normalmente, no necesita cambiar este script.

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

runcat.sh|bat inicia el proceso del servicio de catálogo eXtreme Scale en el sistema local.

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

El script runcontainer.sh|bat inicia un proceso de servidor de contenedor. Puede ejecutar este script varias veces con nombres de servidor exclusivos especificados para iniciar cualquier número de contenedores. Estas instancias pueden trabajar juntas para alojar información con particiones y redundante de la cuadrícula.

- UNIX Linux ./runcontainer.sh *nombre_servidor_exclusivo*
- Windows runcontainer.bat *nombre_servidor_exclusivo*

El script runclient.sh|bat ejecute el cliente de CRUD sencillo e inicia la operación determinada.

- UNIX Linux ./runclient.sh *mandato valor1 valor2*
- Windows runclient.sh *mandato valor1 valor2*

Para *mandato*, utilice una de las siguientes opciones:

- Especifique como i para insertar el *valor2* en la cuadrícula con la clave *valor1*
- Especifique como u para actualizar el objeto con clave de *valor1* a *valor2*

- Especifique como `d` para suprimir el objeto con clave por *valor1*
- Especifique como `g` para recuperar y visualizar el objeto con clave por *valor1*

Nota: El archivo *installRoot*`/ObjectGrid/ gettingstarted/src/Client.java` es el programa cliente que demuestra cómo conectarse a un servidor de catálogo, obtener una instancia de ObjectGrid y utiliza la API ObjectMap.

Pasos básicos

Utilice los siguientes pasos para iniciar la primera cuadrícula y ejecutar un cliente para interactuar con la cuadrícula.

1. Abra una ventana de sesión de terminal o de línea de mandatos.
2. Utilice el siguiente mandato para ir hasta el directorio `gettingstarted`:

   `cd` *installRoot*`/ObjectGrid/gettingstarted`

   Sustituya *installRoot* por la vía de acceso del directorio raíz de instalación de eXtreme Scale o la vía de acceso del archivo raíz de la versión de prueba de eXtreme Scale extraída *installRoot*.
3. Establezca o exporta la variable de entorno JAVA_HOME para hacer referencia a un directorio de instalación válido de JDK o JRE versión 1.5 o posterior.

   UNIX Linux export JAVA_HOME=*directorio_inicio_Java*

   Windows set JAVA_HOME=*directorio_inicio_Java*
4. Ejecute el siguiente script para iniciar un proceso de servicio de catálogo en el sistema principal local:
   - UNIX Linux `./runcat.sh`
   - Windows `runcat.bat`

   El proceso de servicio de catálogo se ejecuta en la ventana actual de terminal.
5. Abra otra ventana de sesión de terminal o de línea de mandatos, y ejecute el siguiente mandato para iniciar una instancia de servidor de contenedor:
   - UNIX Linux `./runcontainer.sh server0`
   - Windows `runcontainer.bat server0`

   El servidor de contenedor se ejecuta en la ventana actual del terminal. Puede repetir el paso 5 y 6 si desea iniciar más instancias de servidor de contenedor para soportar la réplica.
6. Abra otra ventana de sesión de terminal o de línea de mandatos para ejecutar los mandatos de cliente.
   - Añada datos a la cuadrícula:
     - UNIX Linux `./runclient.sh i key1 helloWorld`
     - Windows `runclient.bat i key1 helloWorld`
   - Busque y visualice el valor:
     - UNIX Linux `./runclient.sh g key1`
     - Windows `runclient.bat g key1`
   - Actualice el valor:
     - UNIX Linux `./runclient.sh u key1 goodbyeWorld`
     - Windows `runclient.bat u key1 goodbyeWorld`

- Suprima el valor:
  - UNIX Linux `./runclient.sh d key1`
  - Windows `runclient.bat d key1`

7. Utilice <ctrl+c> para detener el proceso del servicio de catálogo y los servidores de contenedor en las ventanas respectivas.

Definición de un ObjectGrid

El ejemplo utiliza los archivos `objectgrid.xml` y `deployment.xml` que están en el directorio *installRoot*`/ObjectGrid/gettingstarted/xml` para iniciar un servidor de contenedor. El archivo `objectgrid.xml` es el archivo XML de descriptor de ObjectGrid y el archivo `deployment.xml` es el archivo XML de descriptor de política de despliegue de ObjectGrid. Ambos archivos definen de forma conjunta una topología de ObjectGrid distribuida.

Archivo XML de descriptor ObjectGrid

Se utiliza un archivo XML de descriptor de ObjectGrid para definir la estructura del ObjectGrid que es utilizado por la aplicación. Incluye una lista de configuraciones de BackingMap. Estas BackingMaps son el almacenamiento real de datos para los datos almacenados en memoria caché. El ejemplo siguiente es un archivo `objectgrid.xml` de ejemplo. Las primeras líneas del archivo incluyen la cabecera necesaria para cada archivo XML de ObjectGrid. Este archivo de ejemplo define el ObjectGrid Grid con las BackingMaps Map1 y 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>
```

Archivo XML de descriptor de la política de despliegue

Se pasa un archivo XML de descriptor de la política de despliegue a un servidor de contenedor ObjectGrid durante el arranque. Se debe utilizar una política de despliegue con un archivo XML de ObjectGrid y debe ser compatible con el XML de ObjectGrid que se utiliza con la misma. Para cada elemento objectgridDeployment de la política de despliegue, debe tener un elemento ObjectGrid correspondiente en el XML de ObjectGrid. Los elementos backingMap que están definidos dentro del elemento objectgridDeployment deben ser coherentes con las backingMaps que se encuentran en el XML de ObjectGrid. Debe hacerse referencia a cada backingMap dentro de únicamente un mapSet.

El archivo XML de descriptor de política de despliegue intenta emparejarse con el XML correspondiente de ObjectGrid, el archivo `objectgrid.xml`. En el siguiente ejemplo, las primeras líneas del archivo `deployment.xml` incluyen la cabecera necesaria para cada archivo XML de política de despliegue. El archivo define el elemento objectgridDeployment para el ObjectGrid Grid que está definido en el archivo `objectgrid.xml`. Ambas BackingMaps, Map1 y Map2, que están definidas dentro del ObjectGrid Grid se incluyen en el mapSet mapSet que tiene los atributos numberOfPartitions, minSyncReplicas y maxSyncReplicas configurados.

```
<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>
```

El atributo numberOfPartitions del elemento mapSet especifica el número de particiones para el mapSet. Es un atributo opcional y el valor predeterminado es 1. El número debe ser apropiado para la capacidad anticipada de la cuadrícula.

El atributo minSyncReplicas de mapSet especifica el número mínimo de réplicas síncronas para cada partición del mapSet. Se trata de un atributo opcional y el valor predeterminado es 0. El primario y la réplica no se colocan hasta que el dominio pueda soportar el número mínimo de réplicas síncronas. Para dar soporte al valor minSyncReplicas, es necesario un contenedor más que el valor de minSyncReplicas. Si el número de réplicas síncronas cae por debajo del valor de minSyncReplicas, ya no se permiten transacciones de grabación para esa partición.

El atributo maxSyncReplicas de mapSet especifica el número máximo de réplicas síncronas para cada partición del mapSet. Se trata de un atributo opcional y el valor predeterminado es 0. No se coloca ninguna otra réplica síncrona para una partición después de que un dominio alcance este número de réplicas síncronas para dicha partición específica. La adición de contenedores que puedan dar soporte a este ObjectGrid puede comportar un aumento en el número de réplicas síncronas si todavía no se ha alcanzado el valor de maxSyncReplicas. El ejemplo de valor maxSyncReplicas establecido en 1 significa que el dominio colocará, como mínimo, una réplica síncrona. Si inicia más de una instancia de servidor de contenedor, sólo habrá una réplica síncrona colocada en una de las instancias del servidor de contenedor.

Utilización de ObjectGrid

El archivo `Client.java` en el directorio *`installRoot`*`/ObjectGrid/gettingstarted/src/` es el programa cliente que demuestra cómo conectarse al servidor de catálogo, obtener la instancia de ObjectGrid y utilizar la API ObjectMap.

Desde la perspectiva de una aplicación cliente, el uso de WebSphere eXtreme Scale se puede dividir en los pasos siguientes.

1. Conexión al servicio de catálogo obteniendo una instancia de ClientClusterContext.
2. Obtención de una instancia ObjectGrid cliente.
3. Obtención de una instancia Session.
4. Obtención de una instancia ObjectMap.
5. Uso de los métodos ObjectMap.

1. Conexión al servicio de catálogo obteniendo una instancia de ClientClusterContext

Para conectarse al servidor de catálogo, utilice el método connect de la API ObjectGridManager. El método connect utilizado por este ejemplo sólo requiere el punto final de servidor de catálogo en el formato de `nombrehost:puerto`, como, por

ejemplo, `localhost:2809`. Si la conexión con el servidor de catálogo se establece correctamente, el método connect devuelve una instancia de ClientClusterContext. La instancia de ClientClusterContext es necesaria para obtener el ObjectGrid de la API ObjectGridManager. El siguiente fragmento de código demuestra cómo conectarse a un servidor de catálogo y obtener una instancia de ClientClusterContext.

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

2. Obtención de una instancia de ObjectGrid

Para obtener una instancia de ObjectGrid, utilice el método getObjectGrid de la API ObjectGridManager. El método getObjectGrid requiere tanto la instancia de ClientClusterContext, como el nombre de la instancia de ObjectGrid. La instancia de ClientClusterContext se obtiene durante la conexión con el servidor de catálogo. El nombre del ObjectGrid es "Grid" que se especifica en el archivo `objectgrid.xml`. El siguiente fragmento de código demuestra cómo obtener el ObjectGrid llamando al método getObjectGrid del API ObjectGridManager.

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

3. Obtención de una instancia de Session

Puede obtener una Session desde la instancia de ObjectGrid obtenida. Es necesaria una instancia de Session para obtener la ObjectMap, y realizar la demarcación de la transacción. El siguiente fragmento de código demuestra cómo obtener la Session llamando al método getSession de la API ObjectGrid.

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

4. Obtención de una instancia de ObjectMap

Después de obtener una Session, podrá obtener una ObjectMap desde una Session llamando al método getMap de la API Session. Debe pasar el nombre de la correlación como parámetro en el método getMap para poder obtener la ObjectMap. El siguiente fragmento de código demuestra cómo obtener la ObjectMap llamando al método getMap de la API Session.

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

5. Uso de los métodos ObjectMap

Después de que se obtenga un ObjectMap, puede utilizar la API ObjectMap. Recuerde que la ObjectMap es una correlación transaccional y requiere la demarcación de transacción utilizando los métodos begin y commit de la API Session. Si no hay ninguna demarcación de transacción explícita, las operaciones de ObjectMap se ejecutan con transacciones de confirmación automática.

El siguiente fragmento de código demuestra cómo utilizar la API ObjectMap con la transacción de confirmación automática.

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

El siguiente fragmento de código demuestra cómo utilizar la API ObjectMap con la demarcación de transacción explícita.

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

Información adicional

Este ejemplo demuestra cómo iniciar el servidor de catálogo y el servidor de contenedor y cómo utilizar la API ObjectMap en un entorno autónomo. También puede utilizar la API EntityManager.

En un entorno WebSphere Application Server con WebSphere eXtreme Scale instalado o habilitado, el escenario más común es una topología conectada a red. En una topología conectada a red, el servidor de catálogo se encuentran en el proceso del gestor de despliegue de WebSphere Application Server y cada instancia de WebSphere Application Server aloja un servidor eXtreme Scale automáticamente. Las aplicaciones Java™ Platform, Enterprise Edition sólo deben incluir el archivo XML de descriptor de ObjectGrid y, también, el archivo XML de descriptor de la política de despliegue de ObjectGrid en el directorio `META-INF` de cualquier módulo y el ObjectGrid pasa a estar disponible de forma automática. Una aplicación se puede conectar a una servidor de catálogo disponible de forma local y obtener una instancia de ObjectGrid para utilizar.

Capítulo 2. Planificación del despliegue de la aplicación

Antes de desplegar las aplicaciones en WebSphere eXtreme Scale, debe revisar los requisitos de hardware y software, los valores de red y ajuste, las configuraciones de despliegue, etc. También puede utilizar la lista de comprobación operacional para asegurarse de que el entorno está preparado para tener una aplicación desplegada.

Configuraciones de despliegue para eXtreme Scale

WebSphere eXtreme Scale se puede desplegar en dos tipos de entornos: local o distribuido. La configuración necesaria depende del tipo de entorno de despliegue.

Entornos locales

Cuando realiza un despliegue en un entorno local, eXtreme Scale se ejecuta dentro de una única máquina virtual Java, y no se replica. Un entorno local requiere un archivo XML ObjectGrid o las API ObjectGrid.

Entornos distribuidos

Cuando realiza un despliegue en un entorno distribuido, eXtreme Scale se ejecuta en un conjunto de Máquinas virtuales Java . Con esta configuración, puede utilizar el particionamiento y la réplica de datos. Un entorno distribuido se puede clasificar como una topología de despliegue dinámico, ya que se pueden añadir servidores según precise. Igual que en el entorno local, en el entorno distribuido se necesita un archivo XML ObjectGrid, o una configuración equivalente mediante programa. De forma adicional, debe proporcionar un archivo XML de política de despliegue con los detalles de configuración ara la cuadrícula de datos en memoria de eXtreme Scale.

Requisitos de hardware y software

No es necesario utilizar un nivel específico de hardware o sistema operativo para utilizar WebSphere eXtreme Scale. Las sentencias oficiales de soporte para eXtreme Scale se proporcionan en línea en la página Requisitos del sistema.

Si desea una lista detallada de las opciones de hardware y software soportadas por el sistema operativo para WebSphere eXtreme Scale, consulte la página Requisitos del sistema.

Si existe un conflicto entre la información proporcionada en el centro de información y la información en la página Requisitos del sistema, la información del sitio web tienen preferencia. La información de requisitos previos en el centro de información sólo se proporciona por comodidad.

Requisitos de hardware

WebSphere eXtreme Scale no requiere un nivel específico de hardware. Los requisitos de hardware dependen del hardware soportado para la instalación de Java Platform, Standard Edition que utiliza para ejecutar WebSphere eXtreme Scale. Si utiliza eXtreme Scale con WebSphere Application Server u otra implementación de Java Platform, Enterprise Edition, los requisitos de hardware de estas

plataformas son suficientes para WebSphere eXtreme Scale.

Requisitos de sistema operativo

eXtreme Scale no requiere un nivel específico de sistema operativo. Cada implementación de Java SE y Java EE requiere niveles o arreglos distintos de sistema operativo para problemas que se han descubierto durante la comprobación de la implementación de Java. Los niveles necesarios para estas implementaciones son suficientes para eXtreme Scale.

Requisitos de WebSphere Application Server

Los clientes y servidores de eXtreme Scale se ejecutan en un entorno distribuido y los ObjectGrids de memoria local están soportados en WebSphere Application Server versión 6.0.2 y posterior.

Nota: Para utilizar el proveedor de memoria caché dinámica, el sistema debe cumplir uno de los siguientes requisitos mínimos:

- WebSphere Application Server versión 6.1.0.25 o superior + arreglo temporal PK85622
- WebSphere Application Server versión 7.0.0.3 o superior + arreglo temporal PK85622

Consulte los Arreglos recomendados para WebSphere Application Server si desea más información.

Requisitos de otros servidores de aplicaciones

Otras implementaciones de Java EE pueden utilizar el tiempo de ejecución de eXtreme Scale como una instancia local o como un cliente para los servidores eXtreme Scale. Para implementar Java SE, debe utilizar la versión 1.4.2 o posterior.

Ajuste del rendimiento

Para mejorar el rendimiento, considere determinados elementos como, por ejemplo, el ajuste de la red y del sistema operativo, la panificación de puertos de red el ajuste de la maquina virtual Java (JVM) para los valores de WebSphere eXtreme Scale, la configuración de la migración tras error y otros temas de ajustes.

Planificación de puertos de red

WebSphere eXtreme Scale es una memoria caché distribuida que requiere abrir puertos para comunicarse con el intermediario de solicitud de objetos (ORB) y la pila del protocolo de control de transmisiones (TCP) junto con las máquinas virtuales Java y otras máquinas. Debe planificar y controlar los puertos, especialmente, en un entorno de cortafuegos. Por ejemplo, cuando se utilizan contenedores y servicio de catálogo para abrir varios puertos.

Cuadrícula de servicio de catálogo

Una cuadrícula de servicio de catálogo requiere lo siguiente.

1. peerPort para el gestor de alta disponibilidad (HA) para comunicarse entre servidores de catálogo iguales sobre una pila TCP
2. clientPort para los servidores de catálogo para acceder a los datos de servicio de catálogo
3. JMXServicePort para especificar qué puerto debe utilizar el servicio JMX

4. El puerto de escucha ORB para los contenedores y clientes para comunicarse con el servicio de catálogo a través del ORB

Utilice el mandato startOgServer para especificar los puertos listados previamente con la opción en autónomo, tal como se muestra en el siguiente ejemplo:

```
 –catalogServiceEndPoints cs1:host1:clientPort:peerPort, cs2:host2:clientPort:peerPort,
cs3:host3:clientPort:peerPort -listenerPort <orbPort> -JMXServicePort <jmxPort>
```

En un entorno de WebSphere Application Server, los puertos de la lista previa se especifican en las propiedades personalizadas de la célula WebSphere con la clave catalog.services.cluster como:

- cell\node1\server1:host1:clientPort:peerPort:listenerPort
- cell\node2\server2:host2:clientPort:peerPort:listenerPort

Los servidores de contenedor WebSphere eXtreme Scale también requieren varios puertos para funcionar. De forma predeterminada, el servidor de contenedor eXtreme Scale genera su puerto de gestor HA y puerto de escucha ORB automáticamente con los puertos dinámicos. En el entorno de cortafuegos, puesto que es ventajoso para el usuario planificar y controlar los puertos, las opciones se proporcionan para iniciar los servidores de contenedor de eXtreme Scale con el puerto HAManager especificado y el puerto de escucha ORB con una opción en el mandato startOgServer, tal como se indica en el siguiente ejemplo:

```
-HaManagerPort <peerPort>
-listenerPort <orbPort>
```

Una planificación correcta del control de puerto es una ventaja, pero hay una dificultad inherente para planificar y gestionar estos puertos cuando se inician cientos de máquinas virtuales Java en una máquina. Cualquier conflicto de puerto provocará una anomalía en el inicio del servidor.

Cuando la seguridad está habilitada, es necesario un puerto SSL (Secure Socket Layer) además de los puertos listados previamente. Utilizando `Dcom.ibm.CSI.SSLPort=<sslPort>` como un argumento -jvmArgs establece el puerto SSL en `<sslPort>`.

Sistemas operativos y ajuste de red

El ajuste de red puede reducir el retardo de la pila del protocolo de control de transmisiones (TCP) modificando los valores de la conexión activa y mejorar el rendimiento modificando los almacenamientos intermedios de TCP.

Sistemas operativos

Un sistema Windows® necesita menos ajustes, mientras que un sistema Solaris necesita más ajustes. La siguiente información pertenece a cada sistema especificado y podría mejorar el rendimiento de WebSphere eXtreme Scale. Deberá realizar los ajustes de acuerdo con su red y su carga de aplicaciones.

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
```

Propiedades de ORB y valores del descriptor de archivo

Las consideraciones de ajuste incluyen las propiedades del intermediario de solicitud de objetos (ORB) y los valores del descriptor de archivo.

Propiedades ORB

El ORB es utilizado por WebSphere eXtreme Scale para comunicarse en una pila TCP. El archivo `orb.properties` necesario está en el directorio `java/jre/lib`. Para una carga pesada de objetos grandes, habilite la fragmentación ORB especificando el siguiente valor:

```
com.ibm.CORBA.FragmentSize=<right size>
```

Impida el crecimiento de ThreadPool especificando el siguiente valor:

```
com.ibm.CORBA.ThreadPool.IsGrowable=false
```

Establezca los tiempos de espera adecuados para evitar que haya demasiadas hebras en una situación anormal especificando los siguientes valores:

- com.ibm.CORBA.RequestTimeout
- com.ibm.CORBA.ConnectTimeout
- com.ibm.CORBA.FragmentTimeout

Descriptor de archivo

En los sistemas UNIX® y Linux® hay un límite respecto al número de archivos abiertos permitidos por proceso. El sistema operativo especifica el número de

archivos abiertos permitidos. Si este valor se ha establecido en un valor demasiado bajo, se producirá un error de asignación de memoria en AIX y habrá demasiados archivos abiertos y registrados.

En la ventana del terminal del sistema UNIX, establezca este valor en un valor superior al valor del sistema predeterminado. Para grandes máquinas SMP con clones, establezca este valor en ilimitado.

Para las configuraciones de AIX, establezca este valor en `-1` (ilimitado) con el mandato `ulimit -n -1`.

Para las configuraciones Solaris establezca este valor en `16384` con el mandato `ulimit -n 16384`.

Para mostrar el valor actual, utilice el mandato `ulimit –a`.

Ajuste de JVM para WebSphere eXtreme Scale

En esta página se tratan algunas cuestiones específica sobre el ajuste de la máquina virtual Java (JVM) para WebSphere eXtreme Scale. Si tiene problemas de rendimiento, póngase en contacto con el soporte de IBM®. Esta página se actualiza con información de ajuste adicional a medida que vaya estando disponible.

En la mayoría de los casos, WebSphere eXtreme Scale requiere, si los requiere, pocos valores de JVM especiales. Si tiene un gran número de objetos que se están almacenando en WebSphere eXtreme Scale, ajuste el tamaño del almacenamiento dinámico a un nivel apropiado para evitar quedarse sin memoria.

Plataformas

La prueba de rendimiento se ha producido principalmente en máquinas AIX (32 vías), Linux (4 vías), Windows (8 vías) y AZUL (384 vías). Con los sistemas AZUL y las máquinas AIX finales, se pueden enviar escenarios con muchas hebras y los puntos de contienda se pueden identificar y arreglar. Los sistemas AZUL, además, eliminan la recogida de basura, que puede contaminar enormemente las ejecuciones de escalado durante las pruebas.

Requisitos de Object Request Broker (ORB)

IBM SDK incluye una implementación de IBM ORB que se ha probado con WebSphere Application Server y WebSphere eXtreme Scale. Para facilitar el proceso de soporte, utilice una JVM proporcionada por IBM. Otras implementaciones de JVM utilizan un ORB diferente. IBM ORB sólo se proporciona fuera de las cajas de la máquina virtual Java proporcionada por IBM. WebSphere eXtreme Scale requiere un ORB en funcionamiento para poder funcionar. Puede utilizar WebSphere eXtreme Scale con ORB de otros proveedores, pero si se identifica un problema en el ORB, debe ponerse en contacto con el proveedor del ORB para recibir soporte. La implementación del IBM ORB es compatible con las máquinas virtuales Java de otros proveedores y se puede sustituir, si es necesario.

Recogida de basura

WebSphere eXtreme Scale crea objetos temporales asociados a cada transacción como, por ejemplo, una petición y una respuesta y una secuencia de registro. Puesto que estos objetos afectan a la eficacia de la recogida de basura, es muy importante ajustar la recogida de basura.

Para la máquina virtual Java de IBM, utilice el recolector optavgpause para los escenarios con un índice alto de actualizaciones (100% de entradas de modificación de transacciones). El recolector gencon funciona mucho mejor que el recopilador optavgpause para los escenarios en los que los datos se actualizan relativamente con poca frecuencia (un 10% del tiempo o menos). Experimente con los dos tipos de recolectores para ver cuál funciona mejor en su escenario. Si observa un problema de rendimiento, ejecute la operación con la recogida de basura detallada activada para comprobar el porcentaje del tiempo que se emplea en la recogida de basura. Se han dado casos en los que se empleaba el 80% del tiempo en la recogida de basura hasta que se arregló el problema.

Si desea más información sobre cómo configurar la recogida de basura, consulte Ajuste de la máquina virtual Java de IBM.

Rendimiento de la JVM

WebSphere eXtreme Scale se puede ejecutar en distintas versiones de Java 2 Platform, Standard Edition (J2SE). WebSphere eXtreme Scale versión 6.1 soporta J2SE versión 1.4.2 y superior. Para obtener un rendimiento y productividad del programador mayores, utilice J2SE 5 o posterior para aprovechar los beneficios de las anotaciones y de la recogida de basura mejorada. WebSphere eXtreme Scale funciona en máquina virtuales Java de 32 bits o de 64 bits.

WebSphere eXtreme Scale se prueba con un subconjunto de las máquinas virtuales disponibles, sin embargo, la lista soportada no es exclusiva. Puede ejecutar WebSphere eXtreme Scale en cualquier versión 1.4.2 o superior, pero si se identifica un problema en la JVM, debe ponerse en contacto con el proveedor de la JVM para recibir soporte. Si es posible, utilice la JVM del tiempo de ejecución de WebSphere en cualquier plataforma que soporte WebSphere Application Server.

Para la mayoría de los escenarios en los que se utiliza WebSphere eXtreme Scale, Java Platform Standard Edition 6 de la JVM tiene un mejor rendimiento que la edición 5 o 1.4. Java 2 Platform, Standard Edition versión 1.4 tiene un rendimiento bajo, especialmente, para los escenarios que utilizan el recopilador gencon. Java Platform Standard Edition 5 tiene un buen rendimiento, pero el rendimiento de Java Platform, Standard Edition 6 es mejor.

Almacenamientos dinámicos grandes

Cuando la aplicación necesita gestionar una gran cantidad de datos en cada partición, la recogida de basura puede convertirse en un problema. En un escenario donde se realizan básicamente lecturas, el funcionamiento es correcto aunque se utilicen almacenamientos dinámicos grandes (20 GB o más) si se utiliza un recolector generacional. Sin embargo, después de que se rellena el almacenamiento dinámico de tenencia, se produce una pausa proporcional al tamaño del almacenamiento dinámico activo y al número de procesadores de la máquina. Esta pausa puede ser larga en máquinas más pequeñas con almacenamientos dinámicos grandes.

WebSphere eXtreme Scale soporta WebSphere Real Time Java. Con WebSphere Real Time Java, la respuesta del proceso de transacciones para WebSphere eXtreme Scale es más coherente y predecible y el impacto de la recogida de basura y de la planificación de hebras se minimizan de forma significativa. El impacto se reduce hasta el nivel de que la desviación estándar del tiempo de respuesta es menor que el 10% del Java habitual.

Si desea más información, consulte “Utilización de WebSphere Real Time” en la página 222.

Número de hebras

El número de hebras depende de unos pocos factores. Existe un límite en el número de hebras que puede gestionar un solo fragmento. Un fragmento es una instancia de una partición, y puede ser un fragmento primario o de réplica. Con más fragmentos para cada JVM, tendrá más hebras con cada fragmento adicional que proporcionan más vías de acceso simultáneas para los datos. Cada fragmento es tan simultáneo como es posible aunque hay un límite para la simultaneidad.

Configuración de la detección de migración tras error

Puede configurar la cantidad de tiempo entre las comprobaciones de sistema para los servidores que han fallado. Este valor recibe el nombre de intervalo de pulsaciones.

Por qué y cuándo se efectúa esta tarea

La configuración de la migración tras error varía en función del tipo de entorno que utiliza. Si utiliza un entorno autónomo, puede configurar una migración tras error con la línea de mandatos. Si utiliza un entorno WebSphere Application Server Network Deployment, debe configurar la migración tras error en la consola de administración de WebSphere Application Server Network Deployment.

- Configure la migración tras error para los entornos autónomos.

  Puede configurar los intervalos de pulsaciones en la línea de mandatos utilizando el parámetro **-heartbeat** en el archivo de script startOgServer.bat | startOgServer.sh. Establezca este parámetro en uno de los siguientes valores:

Tabla 1. Intervalos de pulsaciones

Valor	Acción	Descripción
0	Típica (valor predeterminado)	Las migraciones tras error se detectan normalmente en 30 segundos.
-1	Agresiva	Las migraciones tras error se detectan normalmente en 5 segundos.
1	Relajada	Las migraciones tras error se detectan normalmente en 180 segundos.

  Un intervalo de pulsaciones agresivo puede ser útil cuando los procesos y la red son estables. Si la red o los procesos no se han configurado de forma óptima, es posible que las pulsaciones se pierdan, lo que comportará en una detección de anomalía falsa.

- Configure la migración tras error para los entornos WebSphere Application Server.

  Puede configurar WebSphere Application Server Network Deployment versión 6.0.2 y posterior para permitir a WebSphere eXtreme Scale que realice la migración tras error muy rápidamente. El tiempo de migración tras error predeterminado para las anomalías severas es aproximadamente de 200 segundos. Una anomalía severa puede ser un fallo grave en la máquina física o servidor, una desconexión del cable de red o un error del sistema operativo. Las anomalías debidas a cuelgues del proceso o a anomalías leves normalmente realizan la migración tras error en menos de un segundo. La detección de

anomalías correspondientes a anomalías leves sucede cuando el sistema operativo cierra automáticamente los sockets de red del proceso inactivo para el servidor que aloja el proceso.

Configuración de pulsaciones de grupo principal

WebSphere eXtreme Scale que se ejecuta en un proceso WebSphere Application Server hereda las características de migración tras error de los valores del grupo principal del servidor de aplicaciones. Las siguientes secciones describen cómo configurar los valores de pulsación del grupo principal para distintas versiones de WebSphere Application Server Network Deployment:

– **Actualice los valores de grupo principal para WebSphere Application Server Network Deployment versión 6.x y 7.x:**

  Especifique el intervalo de pulsación en segundos en las versiones de WebSphere Application Server de la versión 6.0 a la versión 6.1.0.12 o en milisegundos a partir de la versión 6.1.0.13. También debe especificar el número de pulsaciones que faltan. Este valor indica cuántas pulsaciones pueden perderse antes de que se considere anómala una máquina virtual Java (JVM) de igual. El tiempo de detección de anomalías severas es aproximadamente el producto del intervalo de pulsaciones y el número de pulsaciones perdidas.

  Estas propiedades se especifican utilizando las propiedades personalizadas en el grupo principal a través de la consola administrativa de WebSphere. Consulte Propiedades personalizadas del grupo principal para obtener detalles sobre la configuración. Estas propiedades deben especificarse para todos los grupos principales que la aplicación utiliza:

  - El intervalo de pulsaciones se especifica utilizando la propiedad personalizada IBM_CS_FD_PERIOD_SEC para segundos o la propiedad personalizada IBM_CS_FD_PERIOD_MILLIS para milisegundos (requiere V6.1.0.13 o posterior).
  - El número de pulsaciones perdidas se especifica utilizando la propiedad personalizada IBM_CS_FD_CONSECUTIVE_MISSED.

  El valor predeterminado para la propiedad IBM_CS_FD_PERIOD_SEC es 20 y para la propiedad IBM_CS_FD_CONSECUTIVE_MISSED es 10. Si se especifica la propiedad IBM_CS_FD_PERIOD_MILLIS, altera temporalmente cualquier conjunto de propiedades personalizadas IBM_CS_FD_PERIOD_SEC. Los valores de estas propiedades son valores enteros positivos.

  Utilice los siguientes valores para conseguir un tiempo de detección de anomalías de 1500 ms para los servidores WebSphere Application Server Network Deployment versión 6.x:

  - Establezca `IBM_CS_FD_PERIOD_MILLIS = 750` (WebSphere Application Server Network Deployment V6.1.0.13 y posterior)
  - Establezca `IBM_CS_FD_CONSECUTIVE_MISSED = 2`

– **Actualice los valores de grupo principal para WebSphere Application Server Network Deployment versión 7.0**

  WebSphere Application Server Network Deployment versión 7.0 proporciona dos valores de grupo principal que se pueden ajustar para aumentar o reducir la detección de migración tras error:

  - **Periodo de transmisión de pulsación.** El valor predeterminado es 30000 milisegundos.
  - **Periodo de tiempo de espera de pulsación.** El valor predeterminado es 180000 milisegundos.

Si desea más detalles sobre cómo cambiar estos valores, consulte el centro de información de WebSphere Application Server Network Deployment: Valores de descubrimiento y detección de errores.

Utilice los valores siguientes para conseguir un tiempo de detección de anomalías de 1500 ms para los servidores WebSphere Application Server Network Deployment versión 7:

- Establezca el periodo de transmisión de pulsaciones en 750 milisegundos.
- Establezca el periodo de tiempo de espera de pulsaciones en 1500 milisegundos.

Qué hacer a continuación

Cuando estos valores se modifican para proporcionar tiempos de migración tras error cortos, se debe tener en cuenta algunas cuestiones relativas al ajuste del sistema. En primer lugar, Java no es un entorno de tiempo real. Es posible que las hebras se demoren si JVM está sufriendo tiempos de recogida de basura de larga duración. Las hebras también podrían demorarse si la máquina que aloja la JVM tiene mucha carga (debido a la propia JVM o a otros procesos que se ejecutan en la máquina). Si las hebras se retrasan, es posible que las pulsaciones no se envíen a tiempo. En el peor de los casos, podrían demorarse el tiempo de migración tras error necesario. Si las hebras se demoran, se producen detecciones de anomalías falsas. El sistema se debe ajustar y se debe modificar su tamaño para asegurarse de que las detecciones de anomalías falsas no se producen en un entorno de producción. La mejor manera de garantizarlo es utilizando una carga adecuada durante la fase de prueba.

Nota: La versión actual de eXtreme Scale soporta WebSphere Real Time.

Servicio de catálogo de alta disponibilidad

Un servicio de catálogo es la cuadrícula de los servidores de catálogo que utiliza, que conservan la información de topología para todos los contenedores en el entorno de eXtreme Scale. El servicio de catálogo controla el equilibrio de carga y el direccionamiento para todos los clientes. Para desplegar eXtreme Scale como un espacio de proceso de base de datos en memoria, debe agrupar en clúster el servicio de catálogo en una cuadrícula para alta disponibilidad.

Componentes del servicio de catálogo

Cuando se inician varios servidores, uno de ellos se elige como el servidor de catálogo maestro que acepta las pulsaciones IIOP (Internet Inter-ORB Protocol) y maneja los cambios de datos del sistema en respuesta a cualquier cambio de servicio de catálogo o contenedor.

Cuando los clientes se ponen en contacto con uno de los servidores de catálogo, la tabla de direccionamiento de la cuadrícula del servidor de catálogo se propaga en los clientes a través del contexto del servicio CORBA (Common Object Request Broker Architecture).

Configure, como mínimo, tres servidores de catálogo. Si la configuración tiene zonas, puede configurar un servidor de catálogo por zona.

Cuando un servidor o contenedor eXtreme Scale se pone en contacto con uno de los servidores de catálogo, la tabla de direccionamiento de la cuadrícula del servidor de catálogo también se propaga en el servidor y contenedor eXtreme Scale

a través del contexto de servicio CORBA. Además, si el servidor de catálogo contactado no es actualmente el servidor de catálogo maestro, la solicitud se redirecciona automáticamente al servidor de catálogo maestro actual y la tabla de direccionamiento del servidor de catálogo se actualiza.

Nota: Una cuadrícula de servidor de catálogo y una cuadrícula de servidor de contenedor son muy diferentes. La cuadrícula del servidor de catálogo es para la alta disponibilidad de los datos del sistema. La cuadrícula del contenedor es para la alta disponibilidad de datos, la escalabilidad y la gestión de carga de trabajo. Por lo tanto, existen dos tablas de direccionamiento diferentes: la tabla de direccionamiento para la cuadrícula del servidor de catálogo y la tabla de direccionamiento para los fragmentos de la cuadrícula de servidor.

Las responsabilidades del catálogo se dividen en una serie de servicios. El gestor de grupos principales realiza el agrupamiento de igual para la supervisor de estado, el servicio de colocación realiza la asignación, el servicio de administración proporciona acceso a la administración y el servicio de ubicación gestiona la localidad.

Despliegue de la cuadrícula de catálogo

Gestor de grupos principales

El servicio de catálogo utiliza el High Availability Manager (Gestor HA) para agrupar los procesos para la supervisión de la disponibilidad. Cada grupo de procesos es un grupo principal. Con eXtreme Scale, el gestor de grupos principales agrupa dinámicamente los procesos juntos. Estos procesos son pequeños para favorecer la escalabilidad. Cada grupo principal elige un líder que tiene la responsabilidad añadida de enviar el estado al gestor de grupos principales cuando se produce una anomalía en los miembros individuales. Se utiliza el mismo mecanismo de estado para descubrir cuando fallan todos los miembros de un grupo, que provoca que falle la comunicación con el líder.

El gestor de grupos principales es un servicio totalmente automático responsable de organizar los contenedores en pequeños grupos de servidores que se federen automáticamente de manera ligera para conformar un ObjectGrid. Cuando un contenedor se pone en contacto por primera vez con el servicio de catálogo, espera a ser asignado a un grupo nuevo o existente de varios. eXtreme Scale se compone de varios de estos grupos, y este agrupamiento es un habilitador de escalabilidad clave. Cada grupo es un grupo de máquinas virtuales Java que utiliza la pulsación para supervisar la disponibilidad de los otros grupos. Uno de los miembros de estos grupos es elegido líder y tiene la responsabilidad añadida de transmitir información de disponibilidad al servicio de catálogo para permitir reaccionar ante anomalías mediante la reasignación y reenvío de rutas.

Servicio de colocación

El servicio de catálogo gestiona la colocación de fragmentos en el conjunto de contenedores disponibles. El servicio de colocación es responsable de mantener un equilibrio de recursos en los recursos físicos. El servicio de colocación es responsable de asignar fragmentos individuales al contenedor host. Se ejecuta como uno de los N servicios elegidos en la cuadrícula, de forma que siempre hay exactamente una instancia del servicio en ejecución. Si la instancia falla, se elige otro proceso, que tomará el control. El estado del servicio de catálogo se replica en todos los servidores que alojan el servicio de catálogo para favorecer la redundancia.

Administración

El servicio de catálogo es también el punto de entrada lógico para la administración del sistema. EL servicio de catálogo aloja un bean gestionado (MBean) y proporciona los URL de JMX (Java Management Extensions) para cualquiera de los servidores que gestiona el servicio.

Servicio de ubicación

El servicio de ubicación actúa como el punto de contacto para clientes que buscan los contenedores que alojan la aplicación que buscan, y también para los contenedores que registran las aplicaciones alojadas con el servicio de colocación. El servicio de ubicación se ejecuta en todos los miembros de la cuadrícula para ampliar esta función.

El servicio de catálogo contiene lógica que está inactiva durante los estados fijos. Como resultado, el servicio de catálogo influye mínimamente en la escalabilidad. El servicio se crea para dar servicio a cientos de contenedores que pasan a estar disponibles al mismo tiempo. Para la disponibilidad, configure el servicio de catálogo en una cuadrícula.

Planificación

Después de iniciar una cuadrícula de catálogo, los miembros de la cuadrícula se enlazan juntos. Planifique con atención la topología de la cuadrícula del catálogo, porque no podrá modificar la configuración del catálogo durante la ejecución. Extienda la cuadrícula tanto como sea posible para evitar errores.

Inicio de una cuadrícula de servidor de catálogo

Conexión de contenedores eXtreme Scale incorporados en WebSphere Application Server con una cuadrícula de catálogo autónomo

Puede configurar contenedores eXtreme Scale que están incorporados en un entorno WebSphere Application Server para conectarse a una cuadrícula de catálogo autónomo. Utilice la misma propiedad para conectar el servidor de aplicaciones con la cuadrícula de catálogo. No obstante, la propiedad no gestiona el ciclo de vida del catálogo con los servidores. En lugar de esto, la propiedad permite a los contenedores localizar la cuadrícula de catálogo remoto. Para establecer la propiedad, consulte .

Nota: Colisión de nombre de servidor: puesto que esta propiedad se utiliza para iniciar el servidor de catálogo eXtreme Scale así como para conectarse, los servidores de catálogo no deben tener el mismo nombre que ningún servidor WebSphere Application Server.

Consulte la información sobre los quórums del servidor de catálogo en *Visión general del producto* si desea más información.

Lista de comprobación operacional

Utilice la lista de comprobación operacional para preparar el entorno para desplegar WebSphere eXtreme Scale.

Tabla 2. Lista de comprobación operacional

Elemento de lista de comprobación	Para más información

Si utiliza AIX, ajuste los siguientes valores del sistema operativo:
TCP_KEEPINTVL
El valor TCP_KEEPINTVL forma parte de un protocolo de actividad de socket que permite la detección de una caída de red. La propiedad especifica el intervalo entre paquetes que se envían para validar la conexión. Si se utiliza WebSphere eXtreme Scale, establezca el valor en `10`. Para comprobar el valor actual, ejecute el mandato siguiente:
`# no –o tcp_keepintvl`
Para cambiar el valor actual, ejecute el siguiente mandato:
`# no –o tcp_keepintvl=10`
El valor TCP_KEEPINTVL está en medios segundos.
TCP_KEEPINIT
El valor TCP_KEEPINIT forma parte de un protocolo de actividad de socket que permite la detección de una caída de red. La propiedad especifica el valor de tiempo de espera inicial para la conexión TCP. Si se utiliza WebSphere eXtreme Scale, establezca el valor en `40`. Para comprobar el valor actual, ejecute los siguiente mandatos:
`# no –o tcp_keepinit`
Para cambiar el valor actual, ejecute el siguiente mandato:
`# no –o tcp_keepinit=40`
El valor TCP_KEEPINIT está en medios segundos.
	• Si desea la información de ajuste de AIX, consulte Ajuste de los sistemas AIX.

Actualice el archivo `orb.properties` para modificar el comportamiento de transporte de la cuadrícula. El archivo `orb.properties` está en el directorio `java/jre/lib`.	"Archivo de propiedades ORB" en la página 207

Tabla 2. Lista de comprobación operacional (continuación)

Elemento de lista de comprobación	Para más información
Utilice los parámetros del script startOgServer. En particular, utilice los siguientes parámetros: • Establezca los valores de almacenamiento dinámico con el parámetro **-jvmArgs**. • Establezca las classpath y las propiedades de la aplicación con el parámetro **-jvmArgs**. • Establezca los parámetros **-jvmArgs** para configurar la supervisión del agente. **Valores de puerto** WebSphere eXtreme Scale debe abrir los puertos para las comunicaciones para algunos transportes. Estos puertos se han definido todos dinámicamente. Sin embargo, si se utiliza un cortafuegos entre los contenedores, debe especificar los puertos. Utilice la siguiente información sobre los puertos: **Puerto de escucha** Puede utilizar el argumento **-listenerPort** para especificar el puerto que se utiliza para la comunicación entre procesos. **Puerto de grupo principal** Puede utilizar el argumento **-haManagerPort** para especificar el puerto que se utiliza para la detección de anomalías. Tenga en cuenta que los grupos principales no necesitan comunicarse entre zonas, de forma que es posible que no tenga que establecer este puerto, si el cortafuegos está abierto para todos los miembros de una única zona. **Puerto de servicio JMX** Puede utilizar el argumento **-JMXServicePort** para especificar el puerto que debe utilizar el servicio JMX. **Puerto SSL** Pasar `-Dcom.ibm.CSI.SSLPort=1234` como un argumento **-jvmArgs** establece el puerto SSL en `1234`. El puerto SSL es el puerto seguro igual al puerto de escucha. **Puerto de cliente** Sólo se utiliza en el servicio de catálogo. Puede especificar este valor con el argumento **-catalogServiceEndPoints**. El formato del valor de este parámetro está en el formato: `nombreservidor:nombrehost:puertocliente:puertoigual`	"Script startOgServer" en la página 235
Verifique que los valores de seguridad se han configurado correctamente: • Transporte (SSL) • Aplicación (Autenticación y Autorización) Para verificar los valores de seguridad, puede intentar utilizar un cliente dañino para conectarse a la configuración. Por ejemplo, cuando está configurado el valor necesario de SSL, un cliente que tiene un valor TCP_IP con o un cliente con el almacén de confianza erróneo no debe poder conectarse al servidor. Si la autenticación es necesaria, un cliente sin credenciales como, por ejemplo, un ID de usuario y una contraseña, no debe poder conectarse al servidor. Si la autorización se aplica, a un cliente sin autorización de acceso no se le debe otorgar el acceso a los recursos del servidor.	Capítulo 9, "Protección del entorno de despliegue", en la página 309
Elija cómo va a supervisar el entorno. • xsAdmin – Los puertos JMX de los servidores de catálogo deben ser visibles para la herramienta XSAdmin. También se debe poder acceder a los puertos de contenedor para algunos mandatos que recopilan información de los contenedores. • Puede elegir entre las siguientes herramientas de supervisión de proveedor: – Tivoli Enterprise Monitoring Agent – CA Wily Introscope – Hyperic HQ	• "Utilización del programa de utilidad de ejemplo xsAdmin" en la página 287 • "Seguridad JMX (Java Management Extensions)" en la página 320 • "Supervisión con IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale" en la página 291 • "Supervisión de eXtreme Scale con Hyperic HQ" en la página 301 • "Supervisión de aplicaciones de eXtreme Scale con CA Wily Introscope" en la página 297

Capítulo 3. Planificación de la capacidad

Si tiene un tamaño de conjunto de datos inicial y un tamaño de conjunto de datos proyectado, puede planificar la capacidad que necesita para ejecutar WebSphere eXtreme Scale. Aunque dicha planificación le ayuda a desplegar eXtreme Scale de forma eficaz para futuros cambios, lo que le permite maximizar la elasticidad de eXtreme Scale que no tendría con un escenario distinto como, por ejemplo, una base de datos en memoria u otro tipo de base de datos.

Cuadrículas, particiones y fragmentos

Una cuadrícula distribuida de eXtreme Scale se divide en particiones. Una partición contiene un subconjunto exclusivo de los datos. Una partición consta de uno o más fragmentos: un fragmento primario y fragmentos de réplica. No es necesario que tenga fragmentos en una partición, pero los fragmentos de réplica proporcionan alta disponibilidad. Si el despliegue es una cuadrícula de datos de memoria independiente o un espacio de proceso de base de datos de memoria, el acceso de datos en eXtreme Scale se basa en gran medida en los conceptos de la creación de fragmentos.

Los datos de una partición se almacenan en tiempo de ejecución en un conjunto de fragmentos. Este conjunto de fragmentos incluye un fragmento primario y posiblemente uno más fragmentos réplica. Un fragmento es la unidad más pequeña que eXtreme Scale puede añadir a una máquina virtual Java o eliminar de ésta.

Existen dos estrategias de colocación: FIXED_PARTITIONS (valor predeterminado) y PER_CONTAINER. El tema siguiente se centra en el uso de la estrategia FIXED_PARTITIONS.

Número de fragmentos

Si su entorno incluyese diez particiones que contuvieran un millón de objetos sin ninguna réplica, existirían diez fragmentos con 100.000 objetos cada uno. Si añadiera una réplica a este escenario, existiría un fragmento adicional en cada partición. En este caso, existirían 20 fragmentos, 10 fragmentos primarios y 10 fragmentos réplica. De nuevo, cada uno de estos fragmentos contendría 100.000 objetos. Cada partición se compone de un fragmento primario y uno o más (N) fragmentos réplica. La determinación del número óptimo de fragmentos es clave. Si configura un número pequeño de fragmentos, los datos no se distribuyen uniformemente entre los fragmentos, lo que provocará errores de falta de memoria y problemas de sobrecarga del procesador. Debe tener al menos 10 fragmentos para cada JVM al realizar la escala. Al desplegar inicialmente la cuadrícula, debería utilizar posiblemente un número grande de particiones.

Número de fragmentos por JVM

Escenario: número pequeño de fragmentos para cada JVM

Los datos se añaden a una JVM y se eliminan de ésta mediante unidades de fragmentos. Los fragmentos nunca se dividen en partes. Si existen 10 GB de datos y existen 20 fragmentos para alojar estos datos, cada fragmento incluye de media 500 MB de datos. Si hay 9 máquinas virtuales Java que alojan la cuadrícula, de

media cada JVM tiene dos fragmentos. Como 20 no es divisible entre 9, unas cuantas máquinas virtuales Java tienen tres fragmentos, con la distribución siguiente:

- 7 máquinas virtuales Java con 2 fragmentos
- 2 máquinas virtuales Java con 3 fragmentos

Puesto que cada fragmento incluye 500 MB de datos, la distribución de datos no es uniforme. Las 7 máquinas virtuales Java con 2 fragmentos contienen cada uno 1 GB de datos. Las 2 máquinas virtuales Java con 3 fragmentos tienen el 50% más de datos, o 1,5 GB, que es una carga de memoria mucho mayor. Como estas 2 máquinas virtuales Java contienen 3 fragmentos, reciben el 50% más de peticiones de sus datos. Como resultado, un número pequeño de fragmentos para cada JVM causa desequilibrio. Para aumentar el rendimiento, debe incrementar el número de fragmentos para cada JVM .

Escenario: número mayor de fragmentos por JVM

Para este escenario el número de fragmentos es mucho mayor. En este escenario hay 101 fragmentos con 9 máquinas virtuales Java que contienen 10 GB de datos. En este caso, cada fragmento contiene 99 MB de datos. La distribución de fragmentos de las máquinas virtuales Java es la siguiente:

- 7 máquinas virtuales Java con 11 fragmentos
- 2 máquinas virtuales Java con 12 fragmentos

Las 2 máquinas virtuales Java con 12 fragmentos tienen sólo 99 MB más de datos que los otros fragmentos, que es una diferencia del 9%. Este escenario se distribuye de manera más uniforme que la diferencia del 50% del escenario con un número pequeño de fragmentos. Desde la perspectiva de uso del procesador, sólo existe el 9% más de trabajo para las 2 máquinas virtuales Java con los 12 fragmentos en comparación con las 7 máquinas virtuales Java que tienen 11 fragmentos. Al incrementar el número de fragmentos en cada JVM , los datos y el uso del procesador se distribuyen de manera más equilibrada y uniforme.

Al crear el sistema, use 10 fragmentos para cada JVM como escenario de tamaño máximo, o cuando el sistema está ejecutando su número máximo de máquinas virtuales Java en el horizonte de planificación.

Dimensionamiento de la memoria y cálculo del número de particiones

Puede calcular la cantidad de memoria y particiones necesarias para la configuración.

WebSphere eXtreme Scale almacena datos dentro del espacio de direcciones de máquinas virtuales Java (JVM). Cada JVM proporciona espacio de procesador para atender a llamadas para crear, recuperar, actualizar y suprimir, para los datos que están almacenados en la JVM . Además, cada JVM proporciona espacio de memoria para réplicas y entradas de datos. Los objetos Java varían en tamaño, por lo tanto, debe realizar una medición para calcular la cantidad de memoria necesaria.

Para calcular el tamaño de la memoria necesaria, cargue los datos de aplicación en una sola JVM . Cuando el uso del almacenamiento dinámico alcanza el 60%, anote el número de objetos que se utilizan. Este número máximo de objetos recomendado para cada una de las máquinas virtuales Java. Para obtener el tamaño más preciso, utilice datos realistas e incluya todos los índices definidos en el tamaño porque los

índices también consumen memoria. El mejor método para medir el uso de la memoria es ejecutar la salida verbosegc de la recogida de basura porque esta salida le proporciona los números después de la recogida de basura. Puede consultar el uso del almacenamiento dinámico en cualquier punto determinado a través de los MBeans o a través de programa, pero estas consultas sólo le proporcionan una instantánea actual del almacenamiento dinámico, que podría incluir la basura no recopilada, así que utilizar dicho método no es una indicación precisa de la memoria consumida.

Dimensionamiento de la configuración

Número de fragmentos por partición (valor de numShardsPerPartition)

Para calcular el número de fragmentos por partición, o el valor de numShardsPerPartition, añada 1 para el fragmento primario además del número total de fragmentos de réplica que desea.

```
numShardsPerPartition = 1 + número_total_de_réplicas
```

Número de máquinas virtuales Java (valor minNumJVMs)

Para dimensionar la configuración, primero decida sobre el número máximo de objetos que es necesario almacenar en total. Para determinar el número de máquinas virtuales Java que necesita, utilice la siguiente fórmula:

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

Redondee este valor al valor entero más cercano.

. Redondee hasta el entero más cercano.

Número de fragmentos (valor de numShards)

En el tamaño de aumento final, se deben utilizar 10 fragmentos para cada JVM . Tal como se ha descrito anteriormente, cada JVM tiene un fragmento primario y (N-1) fragmentos para las réplicas, o en este caso, 9 réplicas. Puesto que ya tiene un número de máquinas virtuales Java para almacenar los datos, puede multiplicar el número de máquinas virtuales Java por 10 para determinar el número de fragmentos:

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

Número de particiones

Si una partición tiene un fragmento primario y un fragmento de réplica, la partición tiene dos fragmentos (primario y réplica). El número particiones es el total de fragmentos dividido por 2, redondeado por arriba hasta el número primo más cercano. Si la partición tiene un fragmento primario y dos réplicas, el número de particiones es el total de fragmentos dividido por 3, redondeado por arriba hasta el número primo más cercano.

```
numPartitions = numShards / numShardsPerPartition
```

Ejemplo de dimensionamiento

En este ejemplo, el número de entradas empieza en 250 millones. Cada año, el número de entradas aumenta aproximadamente un 14%. Después de 7 años, el número total de entradas es 500 millones, así que debe planificar la capacidad en consecuencia. Para alta disponibilidad, es necesario una sola réplica. Con una

réplica, el número de entradas se dobla, o mil millones de entradas. Como prueba, dos millones de entradas pueden almacenarse en cada JVM . Si se utilizan los cálculos en este escenario, es necesaria la siguiente configuración:

- 500 máquinas virtuales Java para almacenar el número final de entradas.
- 5000 fragmentos, que se calculan multiplicando 500 máquinas virtuales Java por 10.
- 2500 particiones, o 2503 como el siguiente número primo más cercado, que se calculan tomando 5000 fragmentos divididos por dos para los fragmentos primario y de réplica.

Inicio de la configuración

Basándose en los cálculos anteriores, deberá empezar con 250 máquinas virtuales Java y crecer hasta 500 máquinas virtuales Java a lo largo de 5 años, lo que le permite gestionar el crecimiento incremental hasta que llegue al número final de entradas.

En esta configuración, se almacenan alrededor de 200.000 entradas por partición (500 millones de entradas divididas por 2503 particiones). Debe establecer el parámetro numberOfBuckets en la correlación que mantiene las entradas en el número primo más alto más cercado, en este ejemplo 70887, que mantiene el ratio en aproximadamente 3.

Cuando se alcanza el máximo número de máquinas virtuales Java

Cuando se alcanza el máximo número de 500 máquinas virtuales Java, la cuadrícula puede seguir creciendo. Como el número de máquinas virtuales Java aumentas hasta superar 500, el total de fragmentos empieza a caer por debajo de 10 para cada JVM , que está por debajo del número recomendado. Los fragmentos empiezan a crecer, lo que puede causar problemas. Debe repetir el proceso de dimensionamiento teniendo en cuenta el crecimiento en el futuro y restablecer el número de particiones. Este procedimiento requiere un reinicio completo de la cuadrícula, o una parada de la cuadrícula.

Número de servidores

Atención: No utilice la transferencia de páginas en un servidor en ninguna circunstancia.

Una sola JVM utiliza más memoria que el tamaño de almacenamiento dinámico. Por ejemplo, 1 GB de almacenamiento dinámico para una JVM en realidad utiliza 1,4 GB de memoria real. Determine la RAM libre disponible en el servidor. Divida la cantidad de RAM por la memoria por JVM para obtener el número máximo de máquinas virtuales Java en el servidor.

Tamaño de CPU por partición en transacciones

Aunque una funcionalidad principal de eXtreme Scale es su capacidad de escaladas elásticas, también es importante considerar el dimensionamiento y el ajuste del número ideal de las CPU para escalar.

El coste del procesador incluye lo siguiente:

- Coste de los servicios de las operaciones crear, recuperar, actualizar y eliminar en los clientes.
- Coste de la réplica de otras máquinas virtuales Java.

- Coste de la invalidación.
- Coste de la política de desalojo.
- Coste de la recogida de basura.
- Coste de la lógica de la aplicación.

Máquinas virtuales Java por servidor

Utilice dos servidores e inicie el número máximo de JVM por servidor. Utilice el número de particiones calculadas en el apartado anterior. A continuación, precargue las máquinas virtuales Java con un volumen de datos que quepa en estos dos sistemas. Utilice un servidor independiente como cliente. Ejecute una simulación de transacciones realista en esta cuadrícula de dos servidores.

Para calcular la línea base, intente saturar el uso del procesador. Si no puede, es probable que la red esté saturada. Si la red está saturada, añada más tarjetas de red y disponga las máquinas virtuales Java por turno circular en las diversas tarjetas de red.

Ejecute los sistemas con un uso del procesador del 60%, y mida la velocidad de las transacciones crear, recuperar, actualizar y eliminar. El valor que obtenga proporciona el rendimiento de los dos servidores. Este número se dobla con cuatro servidores, y se vuelve a doblar con ocho servidores, y así sucesivamente. Esta escala presupone que la capacidad de la red y la capacidad del cliente también pueden escalarse.

Como resultado, el tiempo de respuesta de eXtreme Scale debe ser estable a medida que se aumenta el número de servidores. El rendimiento de la transacción se debe ampliar de forma lineal a medida que se añadan sistemas a la cuadrícula.

Dimensionamiento de las CPU para transacciones paralelas

Las transacciones de una sola partición dimensionan el rendimiento de forma lineal a medida que la cuadrícula va creciendo. Las transacciones paralelas son distintas de las transacciones de una sola partición porque afectan a un subconjunto de servidores, que podría ser todos los servidores.

Si una transacción afecta a todos los servidores, el rendimiento se limita al rendimiento del cliente que inicia la transacción o el servidor más lento que resulta afectado. Las cuadrículas más grandes esparcen los datos más y proporcionan más espacio de procesador, memoria, red, etc. No obstante, el cliente debe esperar a que el servidor más lento responda, y el cliente debe hacer uso los resultados de la transacción.

Cuando una transacción afecta a un subconjunto de servidores, M de N servidores obtienen una solicitud. A continuación el rendimiento será N dividido por M veces más rápido que el rendimiento del servidor más lento. Por ejemplo, si tiene 20 servidores y una transacción que afecta a 5 servidores, el rendimiento es 4 veces el rendimiento del servidor más lento de la cuadrícula.

Cuando una transacción paralela finaliza, los resultados se envían a la hebra de cliente que ha iniciado la transacción. Este cliente deberá agregar los resultados con una sola hebra. Este tiempo de agregación aumenta a medida que aumenta el número de servidores afectados por la transacción. No obstante, esta vez depende de la aplicación porque es posible que cada servidor devuelva aun resultado más pequeño a medida que va creciendo la cuadrícula.

Normalmente, las transacciones paralelas afectan a todos los servidores en la cuadrícula porque las particiones se distribuyen de forma uniforme por la cuadrícula. En este caso, el rendimiento se limita al primer caso.

Resumen

Con este dimensionamiento, tiene tres medidas, del modo siguiente.

- Número de particiones.
- Número de servidores necesarios para la memoria que es necesaria.
- Número de servidores necesarios para el rendimiento necesario.

Si necesita 10 servidores para los requisitos de memoria, pero sólo obtiene el 50% del rendimiento necesario debido a la saturación en el procesador, necesitará el doble de servidores.

Para obtener la estabilidad más alta, debe ejecutar los servidores al 60% de la carga de procesador y los almacenamientos dinámicos deJVM al 60% de la carga de almacenamiento dinámico. Los picos de utilización pueden conducir al uso del procesador a un 80–90%, aunque de forma habitual no debe ejecutar los servidores a niveles más altos que éstos.

Capítulo 4. Instalación y despliegue de WebSphere eXtreme Scale

WebSphere eXtreme Scale es una cuadrícula de datos en memoria que puede utilizar para crear particiones, replicar y gestionar de forma dinámica datos de aplicación y la lógica empresarial entre varios servidores.

Antes de empezar

- Establezca como WebSphere eXtreme Scale se adapta a la topología actual. Consulte la visión general de la arquitectura y topología de WebSphere eXtreme Scale en *Visión general del producto* si desea más información.
- Verifique que el entorno cumple los requisitos previos para instalar eXtreme Scale. Si desea más información, consulte "Requisitos de hardware y software" en la página 7.

Por qué y cuándo se efectúa esta tarea

Entornos admitidos

No es necesario que instale y despliegue eXtreme Scale en un nivel específico de sistema operativo. Cada instalación de Java Platform, Standard Edition y Java Platform, Enterprise Edition requiere distintos niveles o arreglos de sistema operativo.

Puede instalar y desplegar el producto en los entornos de Java EE y Java SE. También puede empaquetar el componente de cliente con las aplicaciones Java EE directamente si integrarse con WebSphere Application Server. eXtreme Scale soporta Java Runtime Environment (JRE) versión 1.4.2 y posterior y WebSphere Application Server versión 6.0.2 y posterior.

Instale el eXtreme Scale autónomo

Puede instalar el eXtreme Scale autónomo en un entorno que no contiene WebSphere Application Server o WebSphere Application Server Network Deployment. Con la opción autónoma, defina una nueva ubicación de instalación en la que instalar el servidor eXtreme Scale.

Integre el producto con WebSphere Application Server o WebSphere Application Server Network Deployment

Puede instalar e integrar eXtreme Scale con una instalación existente de WebSphere Application Server o WebSphere Application Server Network Deployment. Puede seleccionar instalar tanto el cliente como el servidor de eXtreme Scale, o puede instalar sólo el cliente.

Cree y aumente perfiles

Cree y aumente los perfiles para utilizar las características eXtreme Scale. Si ejecuta WebSphere Application Server versión 6.1 o versión 7.0, puede utilizar el plug-in de herramienta de gestión de perfiles o el mandato manageprofiles. Si ejecuta WebSphere Application Server versión 6.0.2, debe utilizar el mandato wasprofile para crear y aumentar los perfiles.

Aplique el mantenimiento

Utilice IBM Update Installer versión 7.0.0.4 o posterior para aplicar el mantenimiento en el entorno.

Migración a WebSphere eXtreme Scale versión 7.0

Para actualizar la versión 6.1.0.x a la versión 7.0, fusione cualquier archivo de script de producto modificado con los archivos de script del producto nuevo para mantener los cambios.

Antes de empezar

Verifique que los sistemas cumplen los requisitos mínimos para las versiones de producto que tiene previsto migrar e instalar. Si desea más información, consulte "Requisitos de hardware y software" en la página 7.

Por qué y cuándo se efectúa esta tarea

Fusione los archivos de script de producto modificados con los nuevos archivos de script de producto en el directorio `/bin` para mantener los cambios.

Consejo: Si no modifica los archivos de script que están instalado con el producto, no es necesario completar los pasos de migración siguientes. En lugar de esto, puede actualizar a la versión 7.0 desinstalando la versión anterior e instalando la nueva versión en el mismo directorio.

1. Detenga todos los procesos que utilizan eXtreme Scale.
   - Consulte "Detención de servidores de eXtreme Scale autónomos" en la página 239 para detener todos los procesos que se ejecutan en el entorno autónomo de eXtreme Scale.
   - Consulte Programas de utilidad de programa de utilidad de línea de comandos para detener todos los procesos que se ejecutan en el entorno WebSphere Application Server o WebSphere Application Server Network Deployment.
2. Guarde los scripts modificados del directorio de instalación actual en un directorio temporal.
3. Desinstale el producto.
4. Instale eXtreme Scale versión 7.0. Si desea más información, consulte Capítulo 4, "Instalación y despliegue de WebSphere eXtreme Scale", en la página 27.
5. Fusione los cambios de los archivos en el directorio temporal con los nuevos archivos de script de producto en el directorio `/bin`.
6. Inicie todos los procesos de eXtreme Scale para empezar a utilizar el producto. Si desea más información, consulte Capítulo 7, "Administración del entorno", en la página 225.

Instalación de WebSphere eXtreme Scale autónomo

Puede instalar el WebSphere eXtreme Scale autónomo en un entorno que no contiene WebSphere Application Server o WebSphere Application Server Network Deployment.

Antes de empezar

- Verifique que el directorio de instalación de destino está vacío o que no existe.

Nota: Si existe una versión anterior de eXtreme Scale, o el componente ObjectGrid en el directorio que especifica para instalar la versión 7.0, el producto no se instala. Puede elegir un directorio de instalación diferente o cancelar la instalación. A continuación, desinstale la instalación anterior y vuelva a ejecutar el asistente.

- Para obtener un rendimiento y una capacidad de servicio mejores, descargue e instale un kit de desarrollador de IBM desde developerWorks.

Restricción: Windows Si utiliza hardware de un proveedor independiente, seleccione una de las siguientes opciones para descargar e instalar un kit de desarrollador:

– Descargue un Sun JDK.

– Descargue un JDK o JRE de otro proveedor de software independiente.

Por qué y cuándo se efectúa esta tarea

Cuando instale el producto como autónomo, instale de forma independiente el servidor y el cliente de eXtreme Scale. Los procesos de servidor y de cliente, por lo tanto, acceden a todos los recursos necesarios de forma local. También puede incorporar eXtreme Scale en aplicaciones Java Platform, Standard Edition existentes utilizando scripts y archivos JAR (Java Archive).

La tabla siguiente lista los archivos JAR que se incluyen en la instalación.

Tabla 3. Archivos de tiempo de ejecución en el directorio de instalación `/ObjectGrid/lib`

Nombre de archivo	Entorno	Descripción
`cglib.jar`	Local, cliente y servidor	El archivo `cglib.jar` es leído por la función de programa de utilidad cglib cuando se utiliza la modalidad de copia copiar al escribir y cuando se utiliza EntityManager para rastrear los cambios de una entidad. El archivo se incluye automáticamente en el tiempo de ejecución de servidor cuando se utilizan los scripts proporcionados. Añada este archivo al tiempo de ejecución de cliente o local.
`objectgrid.jar`	Local, cliente y servidor	El archivo `objectgrid.jar` es utilizado por el tiempo de ejecución del servidor de Java Platform, Standard Edition versión 1.4.2 y posterior. El archivo se incluye automáticamente en el tiempo de ejecución de servidor cuando se utilizan los scripts proporcionados.
`ogagent.jar`	Local, cliente y servidor	El archivo `ogagent.jar` contiene las clases de tiempo de ejecución necesarias para ejecutar el agente de instrumentación Java que se utiliza con la API EntityManager.
`ogclient.jar`	Local y cliente	El archivo `ogclient.jar` sólo contiene el tiempo de ejecución local y el tiempo de ejecución de cliente. Puede usar este archivo con Java SE Versión 1.4.2 y posterior.
`wsogclient.jar`	Local y cliente	El archivo `wsogclient.jar` se incluye cuando se instala el producto en un entorno que contiene WebSphere Application Server versión 6.0.2 y posterior. Este archivo sólo contiene los tiempos de ejecución local y de cliente.
`wxsdynacache.jar`	Sólo servidor	El archivo `wxsdynacache.jar` contiene las clases necesarias para utilizar con el proveedor de la memoria caché dinámica. El archivo se incluye automáticamente en el tiempo de ejecución de servidor cuando se utilizan los scripts proporcionados. **Atención:** El archivo está en el directorio `ObjectGrid/dynacache/lib`.

1. Utilice el asistente para completar la instalación. Ejecute el siguiente script para iniciar el asistente:
   - Linux UNIX *raíz_dvd*`/install`
   - Windows *raíz_dvd*`\install.bat`
2. Siga las indicaciones del asistente y pulse **Finalizar** para completar la instalación.

   Nota: El panel de características opcionales lista las características que puede optar por instalar. Sin embargo, las características no se pueden añadir de forma incremental en el entorno del producto después de que se instale el

producto. Si elige no instalar una características con la instalación inicial del producto, debe desinstalar y volver a instalar el producto para añadir la característica.

Qué hacer a continuación

Configure los procesos de la aplicación cliente y los procesos de servidor. Si desea más información, consulte Capítulo 6, "Configuración de WebSphere eXtreme Scale", en la página 63.

Referencia relacionada

"Parámetros de instalación" en la página 52
Especifique los parámetros en la línea de mandatos para personalizar y configurar la instalación del producto.

Utilización de el intermediario para solicitudes de objetos con los procesos WebSphere eXtreme Scale

Puede utilizar WebSphere eXtreme Scale con las aplicaciones que utilizan el intermediario de solicitud de objetos (ORB) directamente en los entornos que no contienen WebSphere Application Server o WebSphere Application Server Network Deployment.

Antes de empezar

Si utiliza el ORB dentro del mismo proceso que eXtreme Scale cuando ejecuta aplicaciones, u otros componentes e infraestructuras, que no están incluidos con eXtreme Scale, es posible que tenga que completar tareas adicionales para asegurarse de que eXtreme Scale se ejecuta correctamente en el entorno.

Por qué y cuándo se efectúa esta tarea

Añada la propiedad ObjectGridInitializer al archivo `orb.properties` para inicializar el uso del ORB en el entorno. Utilice el ORB para habilitar la comunicación entre los procesos eXtreme Scale y otros procesos que están en el entono. El archivo `orb.properties` está en el directorio `java/jre/lib`. Consulte "Archivo de propiedades ORB" en la página 207 si desea descripciones de las propiedades y los valores.

Escriba la siguiente línea y guarde las columnas:

```
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.objectgrid.corba.ObjectGridInitializer
```

Resultados

eXtreme Scale inicializa correctamente el ORB y coexiste con otras aplicaciones para las que el ORB está habilitado.

Para utilizar una versión personalizada del ORB con eXtreme Scale, consulte "Configuración de un intermediario de solicitud de objetos personalizado" en la página 55.

Referencia relacionada

"Archivo de propiedades ORB" en la página 207
El archivo `orb.properties` se utiliza para pasar las propiedades utilizadas por el intermediario de solicitud de objetos (ORB) para modificar el comportamiento de transporte de la cuadrícula.

Propiedades personalizadas del intermediario para solicitudes de objetos

Integración de WebSphere eXtreme Scale con WebSphere Application Server

Puede instalar WebSphere eXtreme Scale en un entorno en el que WebSphere Application Server o WebSphere Application Server Network Deployment está instalado. Puede utilizar las características existentes de WebSphere Application Server o WebSphere Application Server Network Deployment para mejorar las aplicaciones aplicando la capacidad de eXtreme Scale.

Antes de empezar

- Instale WebSphere Application Server or WebSphere Application Server Network Deployment. Consulte Instalación del entorno de servicio de aplicaciones si desea más información.
- En función de la versión que instale, la versión 6.0.x, versión 6.1 o la versión 7.0, aplique el fixpack más reciente para WebSphere Application Server o WebSphere Application Server Network Deployment para actualizar el nivel del producto. Consulte los Fixpacks más recientes para WebSphere Application Server si desea más información.
- Verifique que el directorio de instalación de destino no contiene una instalación existente de eXtreme Scale.
- Detenga todos los procesos que se ejecutan en WebSphere Application Server o WebSphere Application Server Network Deployment. Consulte Programas de utilidad de línea de mandatos si desea más información.

Por qué y cuándo se efectúa esta tarea

Integre eXtreme Scale con WebSphere Application Server o WebSphere Application Server Network Deployment para aplicar las características de eXtreme Scale a las aplicaciones Java Platform, Enterprise Edition. Las aplicaciones Java EE alojan cuadrículas eXtreme Scale y acceden a las cuadrículas utilizando una conexión de cliente.

La siguiente tabla lista los archivos JAR (Java Archive) que se incluyen en la instalación.

Tabla 4. Archivos de ejecución en el directorio de instalación `/lib`

Nombre de archivo	Entorno	Descripción
`cglib.jar`	Local, cliente y servidor	El archivo `cglib.jar` es leído mediante la función de programación cglib cuando se utiliza la modalidad de copia copiar al escribir y cuando se utiliza la API EntityManager para rastrear loas cambios de una entidad.
`ogagent.jar`	Local, cliente y servidor	El archivo `ogagent.jar` contiene las clases de tiempo de ejecución necesarias para ejecutar el agente de instrumentación Java que se utiliza con la API EntityManager.
`wsobjectgrid.jar`	Local, cliente y servidor	El archivo `wsobjectgrid.jar` contiene los tiempos de ejecución de eXtreme Scale local, de cliente y de servidor.
`wsogclient.jar`	Local y cliente	El archivo `wsogclient.jar` se incluye cuando se instala el producto en un entorno que contiene WebSphere Application Server versión 6.0.2 y posterior. Este archivo sólo contiene los tiempos de ejecución local y de cliente.

Tabla 4. Archivos de ejecución en el directorio de instalación /lib (continuación)

Nombre de archivo	Entorno	Descripción
wxsdynacache.jar	Sólo servidor	El archivo wxsdynacache.jar contiene las clases necesarias para utilizar con el proveedor de la memoria caché dinámica.

1. Utilice el asistente para completar la instalación. Ejecute el siguiente script para iniciar el asistente:
   - Linux UNIX *raíz_dvd*/install
   - Windows *raíz_dvd*\install.bat
2. Siga las indicaciones del asistente.

   El panel de características opcionales lista las características que puede optar por instalar. Sin embargo, las características no se pueden añadir de forma incremental en el entorno del producto después de que se instale el producto. Si elige no instalar una características con la instalación inicial del producto, debe desinstalar y volver a instalar el producto para añadir la característica.

   El panel de aumento de perfil lista los perfiles existentes que puede seleccionar para aumentar con las características de eXtreme Scale. Sin embargo, si selecciona perfiles que ya están siendo utilizados, se visualiza un panel de aviso. Para continuar con la instalación, detenga los servidores que están configurados en los perfiles, o bien pulse **Atrás** para eliminar los perfiles de la selección.

Qué hacer a continuación

Si ejecuta WebSphere Application Server versión 6.1 o versión 7.0, puede utilizar el plug-in de herramienta de gestión de perfiles o el mandato manageprofiles. Si ejecuta WebSphere Application Server versión 6.0.2, debe utilizar el mandato wasprofile para crear y aumentar los perfiles.

Despliegue la aplicación, inicie un servicio de catálogos e inicie los contenedores en el entorno WebSphere Application Server. Si desea más información, consulte "Administración de WebSphere eXtreme Scale con WebSphere Application Server" en la página 242.

Referencia relacionada

"Parámetros de instalación" en la página 52
Especifique los parámetros en la línea de mandatos para personalizar y configurar la instalación del producto.

Utilización del plug-in Installation Factory para crear e instalar paquetes personalizados

Utilice el plug-in IBM Installation Factory para WebSphere eXtreme Scale para crear un paquete de instalación personalizado (CIP) o un paquete de instalación integrado (IIP). Un CIP contiene un único paquete de instalación y varios activos opcionales. Un IIP combina una o más paquetes de instalación en un único flujo de trabajo de instalación que diseñe.

Antes de empezar

Antes de crear e instalar paquetes personalizados para eXtreme Scale, en primer lugar, debe descargar los siguientes productos:

- IBM Installation Factory para WebSphere Application Server

- Plug-in IBM Installation Factory para WebSphere eXtreme Scale

Por qué y cuándo se efectúa esta tarea

Mediante Installation Factory, puede crear un CIP combinando un único componente de producto con paquetes de mantenimiento, scripts de personalización y otros archivos. Cuando cree un IIP, agregue componentes individuales o paquetes de instalación en un único paquete de instalación.

Archivo de definición de build

Un archivo de definición de build es un documento XML que especifica cómo crear e instalar un paquete de instalación personalizado (CIP) o un paquete de instalación integrado (IIP). IBM Installation Factory for WebSphere eXtreme Scale lee los detalles del paquete del archivo de definición de build para generar un CIP o un IIP.

Antes de poder crear un CIP o un IIP, debe crear un archivo de definición de build para cada paquete personalizado. El archivo de definición de build describe qué componentes de producto o paquetes de instalación para instalar, la ubicación del CIP o el IIP, los paquetes de mantenimiento para incluir, los scripts de instalación y otros archivos que elija incluir. También puede especificar en el archivo de definición de build para el IIP el orden en el que Installation Factory instalar cada paquete de instalación.

El asistente Definición de build le guía a través del proceso de crear un archivo de definición de build. También puede utilizar el asistente para modificar un archivo de definición de build existente. Cada panel del asistente Definición de build le solicita información sobre un paquete personalizado como, por ejemplo, la identificación del paquete, la ubicación de instalación para la definición del build y la ubicación de instalación para el paquete personalizado. Toda esta información se guarda en el nuevo archivo de definición de build, o se modifican o guardan en un archivo de definición de build existente. Si desea más información, consulte los Paneles del asistente Definición del build del CIP y los Paneles del asistente de definición del build del IIP.

Para crear sólo el archivo de definición de build, puede utilizar la herramienta de la interfaz de línea de mandatos para generar el paquete personalizado fuera de la GUI. Si desea más información, consulte “Instalación silenciosa de un CIP o un IIP” en la página 40.

Creación y generación de archivo y generación de un CIP

El plug-in IBM Installation Factory para WebSphere eXtreme Scale genera un paquete de instalación personalizado (CIP) de acuerdo con los detalles que especifique en el archivo de definición de build. La definición de build especifica el paquete de producto para instalar, la ubicación del CIP, los paquetes de mantenimiento para incluir en la instalación, los archivos de script de instalación y cualquier archivo adicional para incluir en el CIP.

Por qué y cuándo se efectúa esta tarea

Puede utilizar el asistente Definición de build para crear un archivo de definición de build y generar un CIP.

1. Ejecute el siguiente script desde el directorio *IF_HOME*/bin para iniciar Installation Factory:
   - UNIX Linux `ifgui.sh`

- Windows `ifgui.bat`

Pulse el icono **Nueva definición de build**.

2. Seleccione el producto para incluir en el archivo de definición de build y pulse **Finalizar** para iniciar el asistente Definición de build.
3. Siga las indicaciones del asistente.

   En el panel Instalar y desinstalar scripts, pulse **Añadir scripts...** para llenar la tabla con ningún script de instalación personalizado. Escriba la ubicación de los archivos de script y desactive el recuadro de selección para continuar si se visualiza un mensaje de error. La operación se detiene de forma predeterminada. Pulse **Aceptar** para volver al panel.

Resultados

Ha creado y personalizado el archivo de definición de build, y ha generado el CIP si ha elegido trabajar en la modalidad conectada.

Si el asistente Definición de build no le proporciona la opción para generar el CIP a partir del archivo de definición de build, podrá seguir generándolo ejecutando el script `ifcli.sh|bat` desde el directorio *IF_HOME*`/bin`.

Qué hacer a continuación

Instale el CIP.

Instalación de un CIP:

Simplifique el proceso de instalación del producto instalando un paquete de instalación personalizado (CIP), que es una imagen de instalación de producto única que puede incluir uno o más paquetes de mantenimiento, scripts de configuración y otros archivos.

Antes de empezar

Antes de poder instalar un CIP, debe crear un archivo de definición de build para especificar qué opciones incluir en el CIP. Si desea más información, consulte “Creación y generación de archivo y generación de un CIP” en la página 33.

Por qué y cuándo se efectúa esta tarea

Un CIP combina e instala un único componente de producto con paquetes de mantenimiento, scripts de personalización y otros archivos.

1. Detenga todos los procesos que se ejecutan en la estación de trabajo que está preparando para la instalación. Para detener el gestor de despliegue, ejecute el siguiente script:
   - Linux UNIX *raíz_perfil*`/bin/stopManager.sh`
   - Windows *raíz_perfil*`\bin\stopManager.bat`

   Para detener los nodos, ejecute el siguiente script:
   - Linux UNIX *raíz_perfil*`/bin/stopNode.sh`
   - Windows *raíz_perfil*`\bin\stopNode.bat`
2. Ejecute el siguiente script para iniciar la instalación:

- Linux UNIX *inicio_CIP*/bin/install
- Windows *inicio_CIP*\bin\install.bat

3. Siga las indicaciones del asistente para completar la instalación.

   El panel de características opcionales lista las características que puede optar por instalar. Sin embargo, las características no se pueden añadir de forma incremental en el entorno del producto después de que se instale el producto. Si elige no instalar una características con la instalación inicial del producto, debe desinstalar y volver a instalar el producto para añadir la característica.

   El panel de aumento de perfil lista los perfiles existentes que puede seleccionar para aumentar con las características de eXtreme Scale. Sin embargo, si selecciona perfiles que ya están siendo utilizados, se visualiza un panel de aviso. Para continuar con la instalación, detenga los servidores que están configurados en los perfiles, o bien pulse **Atrás** para eliminar los perfiles de la selección.

Resultados

Ha instalado correctamente el CIP.

Qué hacer a continuación

Si ejecute WebSphere Application Server versión 6.1 o versión 7.0, puede utilizar el plug-in de Herramienta de gestión de perfiles o el mandato manageprofiles para crear y aumentar perfiles. Si ejecuta WebSphere Application Server versión 6.0.2, debe utilizar el mandato wasprofile para crear y aumentar los perfiles. Si desea más información, consulte “Creación y aumento de perfiles para WebSphere eXtreme Scale” en la página 41.

Si ha aumentado los perfiles para eXtreme Scale durante el proceso de instalación, puede desplegar aplicaciones, iniciar un servicio de catálogo e iniciar los contenedores en el entorno de WebSphere Application Server. Si desea más información, consulte “Administración de WebSphere eXtreme Scale con WebSphere Application Server” en la página 242.

Instalación de un CIP para aplicar el mantenimiento a una instalación del producto existente:

Puede aplicar paquetes de mantenimiento a una instalación de producto existente instalando un paquete de instalación personalizado (CIP). Normalmente se hace referencia al proceso de aplicar el mantenimiento a una instalación existente con un CIP como *instalación de SLIP*.

Antes de empezar

Cree un archivo de definición de build para especificar qué opciones incluir en el CIP. Si desea más información, consulte “Creación y generación de archivo y generación de un CIP” en la página 33.

Por qué y cuándo se efectúa esta tarea

Cuando se aplica el mantenimiento con un CIP que contiene un paquete de renovación, un fixpack, o ambos, el asistente desinstala todos los informes autorizados de análisis de programa (APAR) instalados previamente. Si el CIP está en el mismo nivel que el producto, los APAR instalados previamente se conservan

sólo si se han empaquetado en el CIP. Para aplicar correctamente el mantenimiento a una instalación existente, que es , debe incluir las características instaladas en el CIP.

1. Detenga todos los procesos que se ejecutan en la estación de trabajo que está preparando para la instalación. Para detener el gestor de despliegue, ejecute el siguiente script:
   - Linux UNIX *raíz_perfil*/bin/stopManager.sh
   - Windows *raíz_perfil*\bin\stopManager.bat

   Para detener los nodos, ejecute el siguiente script:
   - Linux UNIX *raíz_perfil*\bin\stopNode.sh
   - Windows *raíz_perfil*\bin\stopNode.bat
2. Ejecute el siguiente script para iniciar la instalación:
   - Linux UNIX *inicio_CIP*/bin/install
   - Windows *inicio_CIP*\bin\install.bat
3. Siga las indicaciones del asistente para completar la instalación.

   El resumen de la vista previa de instalación lista la versión de producto resultante y las características y los arreglos temporales aplicables. A continuación, el asistente aplica correctamente el mantenimiento y actualiza la características del producto.

Resultados

Los archivos binarios del producto se copian en el directorio *inicio_was*/properties/version/nif/backup. Puede utilizar el IBM Update Installer para desinstalar la actualización y restaurar la estación de trabajo. Si desea más información, consulte "Desinstalación de actualizaciones del CIP de una instalación de producto existente".

Desinstalación de actualizaciones del CIP de una instalación de producto existente:

Puede eliminar las actualizaciones del CIP de una instalación de producto existente sin eliminar todo el producto. Utilice IBM Update Installer versión 7.0.0.4 para desinstalar cualquier actualización. También se hace referencia a esta tarea como *desinstalación de SLIP*.

Antes de empezar

Debe tener, como mínimo, una copia existente del producto instalado en el sistema.

1. Descargue la versión 7.0.0.4 del instalador de actualización desde el siguiente sitio FTP:

   ftp://ftp.software.ibm.com/software/websphere/cw/process_server/FEP/UPDI/7004
2. Instale el instalador de actualización. Consulte Instalación del instalador de actualización para el software WebSphere en el centro de información de WebSphere Application Server si desea más información.
3. Desinstale los fixpacks, paquetes de renovación o arreglos temporales que ha añadido en el entorno después de haber instalado el CIP.

4. Desinstale los arreglos temporales que ha incluido en la instalación de SLIP. Este proceso es el mismo que la desinstalación de un único fixpack o paquete de renovación. Sin embargo, el mantenimiento que se incluyó en el CIP ahora se incluye en una sola operación.
5. Desinstale el CIP utilizando el instalador de actualización. Los niveles de mantenimiento lo devuelven al estado previo de la actualización y el CIP se denota a través del identificador del CIP que se incluye como prefijo a su nombre de archivo. El siguiente ejemplo muestra cómo se visualiza un CIP de forma diferente que los otros paquetes de mantenimiento regulares en el panel de selección del paquete de mantenimiento:

   CIP

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

Resultados

Ha eliminado correctamente las actualizaciones del CIP de una instalación de producto existente.

Creación de un archivo de definición de build y generación de un IIP

El plug-in IBM Installation Factory para WebSphere eXtreme Scale genera un IIP basado en las propiedades que proporciona el archivo de definición de build como, por ejemplo, qué paquetes de instalación incluir en el IIP, el orden en el que Installation Factory instala cada paquete y la ubicación del IIP.

Por qué y cuándo se efectúa esta tarea

Puede utilizar el asistente Definición de build para crear un archivo de definición de build y generar un IIP.

1. Ejecute el siguiente script desde el directorio *IF_HOME*/bin para iniciar Installation Factory:
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`
2. Pulse el icono **Crear nuevo paquete de instalación integrado** para iniciar el asistente Definición de build.
3. Siga las indicaciones del asistente.
   a. En el panel Construir el IIP, seleccione un paquete de instalación soportado en la lista y pulse **Añadir instalador** para añadir el paquete de instalación al IIP. Se visualiza un panel que muestra el nombre de paquete, el identificador del paquete y las propiedades del paquete. Para ver información específica sobre el paquete seleccionado, pulse **Ver información de paquete de instalación**. Pulse **Modificar** para especificar la vía de acceso del directorio al paquete de instalación para cada sistema operativo. Si está añadiendo actualmente un paquete de instalación para WebSphere Extended Deployment, active el recuadro de selección, que le proporciona la opción de utilizar el mismo paquete para todos los sistemas operativos soportados. Pulse **Aceptar** y vuelva al panel Construir el IIP. Se crea una invocación de forma predeterminada.
      - Para modificar la vía de acceso del directorio de un paquete de instalación, seleccione el paquete en los paquetes de instalación utilizados en la lista de IIP y pulse **Modificar**.
      - Para modificar una invocación, selecciónela y pulse **Modificar**. Especifique la ubicación de instalación predeterminada para la invocación

en cada sistema operativo. Especifique la ubicación del archivo de respuestas si selecciona una instalación silenciosa como la modalidad de instalación predeterminada.

- Pulse **Añadir invocación** para añadir una contribución de invocación al paquete de instalación. Se visualiza un panel desde el que puede especificar las propiedades para la invocación.
- Pulse **Eliminar** para eliminar los paquetes de instalación o las invocaciones.

4. Revise el resumen de las selecciones, seleccione la opción **Guardar archivo de definición de build y generar el paquete de instalación integrado** y pulse **Finalizar**.

   De forma alternativa, puede optar por guardar el archivo de definición de build sin generar el IIP. Con esta opción, genera realmente el IIP fuera del asistente ejecutando el script `ifcli.bat | ifcli.sh` desde el directorio *inicio_IF*/bin/.

Resultados

Ha creado y personalizado el archivo de definición de build para un IIP.

Qué hacer a continuación

Instale el IIP.

Instalación de un IIP:

Utilice el plug-in IBM Installation Factory para WebSphere eXtreme Scale para instalar un paquete de instalación integrado (IIP). Un IIP combina uno o más paquetes de instalación en una flujo de trabajo único que diseñe.

Antes de empezar

Antes de poder instalar un CIP, debe crear un archivo de definición de build para especificar qué opciones incluir en el CIP. Si desea más información, consulte "Creación de un archivo de definición de build y generación de un IIP" en la página 37.

Por qué y cuándo se efectúa esta tarea

Un IIP puede incluir uno o más paquetes de instalación disponibles de forma general, uno o más CIP y otros archivos y directorios opcionales. Mediante la instalación de un IIP, se agregan varios paquetes de instalación, o *contribuciones*, en un único paquete y, a continuación, se instalan las contribuciones en un orden específico para completar y una instalación completa.

1. Ejecute el siguiente script para iniciar el asistente:
   - Linux UNIX *inicio_IIP*/bin/install
   - Windows *inicio_IIP*\bin\install.bat
2. Pulse **Acerca de** en el panel de bienvenida para ver los detalles del IIP como, por ejemplo, el identificador del paquete, los sistemas operativos soportados y los paquetes de instalación incluidos.

   Opcional: Para modificar las opciones de instalación para cada paquete, pulse **Modificar**.

Opcional: Se visualizan dos botones **Ver registro** en el panel del asistente. Para ver el registro de cada paquete, pulse el botón **Ver registro** que se visualiza junto a la lista que lista los paquetes de instalación. Para ver los detalles generales del registro del IIP, pulse el botón **Ver registro** que se visualiza junto a la información del estado.

3. Seleccione los paquetes de instalación para ejecutar y pulse **Instalar**. Se visualiza una lista de todas las contribuciones en el orden de invocación que contiene el IIP. Para designar qué invocaciones de contribución no se deben ejecutar durante la instalación, desactive el recuadro de selección situado junto al campo **Nombre de instalación**.

Resultados

Ha instalado correctamente un IIP.

Modificación de un archivo de definición de build existente para un IIP:

Puede editar o añadir las propiedades de un IIP para personalizar de forma adicional la instalación.

Por qué y cuándo se efectúa esta tarea

Para cambiar las propiedades de un IIP, modifique el archivo de definición de build existente.

1. Ejecute el siguiente script desde el directorio *IF_HOME*/bin para iniciar Installation Factory:
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`
2. Pulse el icono **Abrir definición de build** y seleccione el archivo de definición de build que desee modificar.
3. Seleccione las propiedades específicas del IIP que desee modificar. La siguiente lista contiene las posibles modificaciones que puede realizar:
   - Cambiar la selección de modalidad actual. En la modalidad conectada, crear la definición del build para utilizar y, de forma opcional, generar el IIP, desde la estación de trabajo actual. En la modalidad desconectada, crear el archivo de definición de build para utilizar en otra estación de trabajo.
   - Añadir o eliminar los sistemas operativos existentes que soporta el IIP.
   - Editar el identificador y la versión existentes para el IIP.
   - Editar la ubicación de destino para el archivo de definición de build.
   - Editar la ubicación de destino para el IIP.
   - Cambiar si se visualiza un asistente de instalación para el IIP. El asistente proporciona información sobre el IIP y las opciones de instalación cuando se ejecuta el IIP.
   - Añadir, eliminar y editar los paquetes de instalación que se incluyen en el IIP.

     Importante: Si ha añadido un sistema operativo soportado y no ha actualizado las propiedades del paquete de instalación en el IIP, recibirá un mensaje de aviso que indica que las contribuciones seleccionadas no contienen paquetes de instalación que se hayan identificado para todos los sistemas operativos que soporta el IIP. Pulse **Sí** para continuar, o pulse **No** para editar el paquete de instalación.

4. Revise el resumen de las selecciones, seleccione **Guardar archivo de definición de build y generar paquete de instalación integrado** y pulse **Finalizar**.

Instalación silenciosa de un CIP o un IIP

Puede instalar de forma silenciosa un paquete de instalación personalizado (CIP) o un paquete de instalación integrado (IIP) para el producto utilizando un archivo de respuestas plenamente cualificado, que configura de forma específica según sus necesidades, o parámetros que pasa a la línea de mandatos.

Antes de empezar

Cree el archivo de definición de build para el CIP o el IIP. Si desea más información, consulte "Creación y generación de archivo y generación de un CIP" en la página 33.

Por qué y cuándo se efectúa esta tarea

Una instalación silenciosa utiliza el mismo programa de instalación que utiliza la versión de la interfaz gráfica de usuario (GUI). Sin embargo, en lugar de visualizar una interfaz de asistente, la instalación silenciosa lee todas las respuestas de un archivo que personaliza, o de los parámetros que pase a la línea de mandatos. Si instala de forma silenciosa un IIP, puede invocar una contribución con una combinación de opciones que especifica directamente en la línea de mandatos, así como las opciones que especifique en un archivo de respuestas. Sin embargo, las opciones de contribución que pase a la línea de mandatos provoca que el instalador del IIP ignore todas las opciones que se especifican en un archivo de respuestas de contribución específica. Consulte las Opciones de instalación de IIP detalladas si desea más información.

Nota: Debe especificar el nombre de archivo de respuestas completo. La especificación de la vía de acceso relativa provoca que la instalación falles sin ninguna indicación de que se haya producido un error.

1. Opcional: Si opta por instalar el CIP o IIP utilizando un archivo de respuestas, en primer lugar, personalice el archivo.
   a. Copie el archivo de respuestas, `wxssetup.response.txt`, del DVD del producto en la unidad de disco.
   b. Abra y edite el archivo de respuestas en el editor de texto que elija. El archivo incluye comentarios para ayudar al proceso de configuración y debe incluir estos parámetros:
      - El acuerdo de licencia
      - La ubicación de la instalación del producto

      Consejo: El instalador utiliza la ubicación que seleccione para la instalación para determinar dónde está instalada la instancia de WebSphere Application Server. Si realiza la instalación en un nodo con varias instancias de WebSphere Application Server, defina de forma clara la ubicación.
   c. Ejecute el siguiente script para iniciar el archivo de respuestas personalizado.
      - Linux UNIX `install -options /vía_acceso_absoluto/archivo_respuesta.txt -silent`
      - Windows `install.bat -options C:\vía_acceso_unidad\archivo_respuesta.txt -silent`

2. Opcional: Si opta por instalar el CIP o IIP pasando determinados parámetros a la línea de mandatos, ejecute el siguiente script para iniciar la instalación:
   - Linux UNIX `install -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=`*`ubicación_instalación`*
   - Windows `install.bat -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=`*`ubicación_instalación`*

   donde *ubicación_instalación* es la ubicación de la instalación existente de WebSphere Application Server.
3. Revise los registros resultantes para ver los errores o una anomalía de instalación.

Resultados

Ha instalado de forma silenciosa el CIP o IIP.

Qué hacer a continuación

Si ejecute WebSphere Application Server versión 6.1 o versión 7.0, puede utilizar el plug-in de Herramienta de gestión de perfiles o el mandato manageprofiles para crear y aumentar perfiles. Si ejecuta WebSphere Application Server versión 6.0.2, debe utilizar el mandato wasprofile para crear y aumentar los perfiles.

Si ha aumentado los perfiles para eXtreme Scale durante el proceso de instalación, puede desplegar aplicaciones, iniciar un servicio de catálogo e iniciar los contenedores en el entorno de WebSphere Application Server. Si desea más información, consulte "Administración de WebSphere eXtreme Scale con WebSphere Application Server" en la página 242.

Creación y aumento de perfiles para WebSphere eXtreme Scale

Después de instalar el producto, cree tipos exclusivos de perfiles y aumente los existentes para WebSphere eXtreme Scale.

Antes de empezar

Instale WebSphere eXtreme Scale. Si desea más información, consulte "Integración de WebSphere eXtreme Scale con WebSphere Application Server" en la página 31.

Por qué y cuándo se efectúa esta tarea

Ejecución en WebSphere Application Server Versión 6.0.2

Si el entorno contiene WebSphere Application Server versión 6.0.2, utilice el mandato wasprofile para crear o aumentar perfiles para WebSphere eXtreme Scale, tal como se muestra en el siguiente ejemplo:

```
raíz_instalación/bin/wasprofile.sh|bat -augment -profileName dmgr_01
-templatePath "C:/Archivos de programa/IBM/WebSphere/AppServer/profileTemplates/xs_augment/dmgr"
```

Consulte el mandato wasprofile en el centro de información de WebSphere Application Server si desea más información.

Ejecución en WebSphere Application Server versión 6.1 o versión 7.0

Si el entorno contiene WebSphere Application Server versión 6.1 o versión 7.0, puede utilizar el plug-in Profile Management Tool o el mandato manageprofiles para crear y aumentar perfiles.

Qué hacer a continuación

En función de la tarea que elija completar, inicie la consola Primeros pasos para recibir ayuda en la configuración y la pruebas del entorno del producto. De forma alternativa, repita cualquier tarea previa para crear o aumentar perfiles adicionales.

Referencia relacionada

"Parámetros de instalación" en la página 52
Especifique los parámetros en la línea de mandatos para personalizar y configurar la instalación del producto.

"Mandato manageprofiles" en la página 44
Puede utilizar el programa de utilidad manageprofiles para crear perfiles con la plantilla de WebSphere eXtreme Scale, y aumentar y reducir los perfiles existentes del servidor de aplicaciones con las plantillas de aumento de eXtreme Scale. Para utilizar las características del producto, el entorno debe contener, como mínimo, un perfil aumentado para el producto.

Utilización de la interfaz gráfica de usuario para crear perfiles

Utilice la interfaz gráfica de usuario (GUI), que proporciona el plug-in de la herramienta de gestión de perfiles para crear perfiles para WebSphere eXtreme Scale. Un perfil es un conjunto de archivos que define el entorno de ejecución.

Antes de empezar

Nota: Si ejecuta WebSphere Application Server versión 6.0.2 o WebSphere Application Server Network Deployment versión 6.0.2, debe utilizar el mandato wasprofile para crear o aumentar un perfil para WebSphere eXtreme Scale tal como se indica en el siguiente ejemplo:

```
raíz_instalación/bin/wasprofile.sh|bat -create -profileName dmgr_01
-templatePath "C:/Archivos de programa/IBM/WebSphere/AppServer/profileTemplates/xs_augment/dmgr"
```

Consulte el mandato wasprofile en el centro de información de WebSphere Application Server si desea más información.

Por qué y cuándo se efectúa esta tarea

Para utilizar las características del producto, el plug-in de Herramienta de gestión de perfiles permite a la GUI ayudarle a configurar perfiles como un perfil de WebSphere Application Server, un perfil de gestor de despliegue, un perfil de célula y un perfil personalizado.

Utilice la GUI de la herramienta de gestión de perfiles para crear perfiles. Elija de una de las opciones siguientes para iniciar el asistente:

- Seleccione **Herramienta de gestión de perfiles** en la consola Primeros pasos.
- Acceda a la herramienta de gestión de perfiles desde el menú **Inicio**.
- Ejecute el script ./pmt.sh|bat desde el directorio *raíz_instalación*/bin/ProfileManagement.

Se visualiza la página Selección de acción sólo si existen, como mínimo, un perfil y las plantillas de aumento.

Qué hacer a continuación

Puede crear perfiles adicionales o aumentar los perfiles existentes. Para reiniciar la herramienta de gestión de perfiles, ejecute el mandato ./pmt.sh|bat desde el directorio *raíz_instalación*/bin/ProfileManagement, o seleccione **Herramienta de gestión de perfiles** en la consola Primeros pasos.

Inicie un servicio de catálogos, inicie contenedores y configure los puertos TCP en el entorno WebSphere Application Server. Si desea más información, consulte "Administración de WebSphere eXtreme Scale con WebSphere Application Server" en la página 242.

Referencia relacionada

"Mandato manageprofiles" en la página 44
Puede utilizar el programa de utilidad manageprofiles para crear perfiles con la plantilla de WebSphere eXtreme Scale, y aumentar y reducir los perfiles existentes del servidor de aplicaciones con las plantillas de aumento de eXtreme Scale. Para utilizar las características del producto, el entorno debe contener, como mínimo, un perfil aumentado para el producto.

Utilización de la interfaz gráfica de usuario para aumentar perfiles

Después de instalar el producto, podrá aumentar un perfil existente para que sea compatible con WebSphere eXtreme Scale.

Antes de empezar

Nota: Si ejecuta WebSphere Application Server versión 6.0.2 o WebSphere Application Server Network Deployment versión 6.0.2, debe utilizar el mandato wasprofile para crear o aumentar un perfil para WebSphere eXtreme Scale tal como se indica en el siguiente ejemplo:

```
raíz_instalación/bin/wasprofile.sh|bat -augment -profileName dmgr_01
-templatePath "C:/Archivos de programa/IBM/WebSphere/AppServer/profileTemplates/xs_augment/dmgr"
```

Consulte el mandato wasprofile en el centro de información de WebSphere Application Server si desea más información.

Por qué y cuándo se efectúa esta tarea

Cuando aumente un perfil existente, cambie el perfil aplicando una plantilla de aumento específica del producto. Por ejemplo, los servidores WebSphere eXtreme Scale no se inician automáticamente, a menos que el perfil de servidor se aumente con la plantilla xs_augment.

- Aumente el perfil con la plantilla xs_augment si ha instalado el cliente eXtreme Scale o el cliente y el servidor.
- Aumente el perfil con la plantilla pf_augment si ha instalado sólo el recurso de particionamiento.
- Aplique ambas plantillas, si el entorno contiene el cliente eXtreme Scale y el recurso de particionamiento.

Utilice la GUI de la herramienta de gestión de perfiles para aumentar los perfiles para eXtreme Scale. Elija de una de las opciones siguientes para iniciar el asistente:

- Seleccione **Herramienta de gestión de perfiles** en la consola Primeros pasos.
- Acceda a la herramienta de gestión de perfiles desde el menú **Inicio**.

- Ejecute el script ./pmt.sh|bat desde el directorio *raíz_instalación*/bin/ProfileManagement.

Qué hacer a continuación

Puede aumentar los perfiles adicionales. Para reiniciar la herramienta de gestión de perfiles, ejecute el mandato ./pmt.sh|bat desde el directorio *raíz_instalación*/bin/ProfileManagement, o seleccione **Herramienta de gestión de perfiles** en la consola Primeros pasos.

Inicie un servicio de catálogos, inicie contenedores y configure los puertos TCP en el entorno WebSphere Application Server. Si desea más información, consulte "Administración de WebSphere eXtreme Scale con WebSphere Application Server" en la página 242.

Referencia relacionada

"Mandato manageprofiles"
Puede utilizar el programa de utilidad manageprofiles para crear perfiles con la plantilla de WebSphere eXtreme Scale, y aumentar y reducir los perfiles existentes del servidor de aplicaciones con las plantillas de aumento de eXtreme Scale. Para utilizar las características del producto, el entorno debe contener, como mínimo, un perfil aumentado para el producto.

Mandato manageprofiles

Puede utilizar el programa de utilidad manageprofiles para crear perfiles con la plantilla de WebSphere eXtreme Scale, y aumentar y reducir los perfiles existentes del servidor de aplicaciones con las plantillas de aumento de eXtreme Scale. Para utilizar las características del producto, el entorno debe contener, como mínimo, un perfil aumentado para el producto.

- Antes de poder crear y aumentar los perfiles, debe instalar eXtreme Scale . Si desea más información, consulte "Integración de WebSphere eXtreme Scale con WebSphere Application Server" en la página 31.
- Si el entorno contiene WebSphere Application Server versión 6.0.2, debe utilizar el mandato wasprofile para crear y aumentar los perfiles para eXtreme Scale tal como se muestra en el ejemplo siguiente:

  ```
  raíz_instalación/bin/wasprofile.sh|bat -augment -profileName dmgr_01
  -templatePath "C:/Archivos de programa/IBM/WebSphere/AppServer/profileTemplates/xs_augment/dmgr"
  ```

  Consulte el mandato wasprofile en el centro de información de WebSphere Application Server si desea más información.

Finalidad

El mandato manageprofiles crea el entorno de ejecución para un proceso de producto en un conjunto de archivos llamado perfil. El perfil define el entorno de ejecución. Puede realizar las siguientes acciones con el mandato manageprofiles:

- Crear y aumentar un perfil de gestor de despliegue
- Crear y aumentar un perfil personalizado
- Crear y aumentar un perfil de servidor de aplicación autónomo
- Crear y aumentar un perfil de célula
- Reducir cualquier tipo de perfil

Cuando aumente un perfil existente, cambie el perfil aplicando una plantilla de aumento específica del producto.

- Aumente el perfil con la plantilla xs_augment si ha instalado el cliente de eXtreme Scale, o el cliente y también el servidor.

- Aumente el perfil con la plantilla pf_augment si ha instalado sólo el recurso de particionamiento.
- Aplique ambas plantillas si el entorno contiene el cliente de eXtreme Scale y el recurso de particionamiento.

Ubicación

El archivo de mandato está en el directorio *raíz_instalación*/bin.

Uso

Si desea ayuda detallada, utilice el parámetro **-help**:

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

En las siguientes secciones, se describen las tareas que puede realizar utilizando el mandato manageprofiles, junto una lista de los parámetros necesarios. Si desea detalles sobre los parámetros opcionales para cada tarea, consulte el mandato manageprofiles en el centro de información de WebSphere Application Server.

Crear un perfil de gestor de despliegue

Puede utilizar el mandato manageprofiles para crear un perfil de gestor de despliegue. El gestor de despliegue administra los servidores de aplicaciones que se han federado en la célula.

Parámetros

-create
Crea un perfil. (Necesario)

-templatePath *vía_acceso_plantilla*
Especifica la vía de acceso de archivo de la plantilla. (Necesario)

Utilice el siguiente formato:

```
-templatePath raíz_instalación/profileTemplates/tipo_plantilla/dmgr
```

donde *tipo_plantilla* es `xs_augment` o `pf_augment`.

Ejemplo

- Utilización de la plantilla xs_augment:

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

- Utilización de la plantilla pf_augment:

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

Crear un perfil personalizado

Puede utilizar el mandato manageprofiles para crear un perfil personalizado. Un perfil personalizado es un nodo vacío que puede personalizar a través del gestor de despliegue para incluir servidores de aplicaciones, clústeres u otros procesos Java.

Parámetros

-create
Crea un perfil. (Necesario)

-templatePath *vía_acceso_plantilla*
Especifica la vía de acceso de archivo de la plantilla. (Necesario)

Utilice el siguiente formato:

```
-templatePath raíz_instalación/profileTemplates/tipo_plantilla/managed
```

donde *tipo_plantilla* es `xs_augment` o `pf_augment`.

Ejemplo

- Utilización de la plantilla xs_augment:

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

- Utilización de la plantilla pf_augment:

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

Crear un perfil de servidor de aplicaciones autónomo

Puede utilizar el mandato manageprofiles para crear un perfil de servidor de aplicaciones autónomo.

Parámetros

-create
Crea un perfil. (Necesario)

-templatePath *vía_acceso_plantilla*
Especifica la vía de acceso de archivo de la plantilla. (Necesario)

Utilice el siguiente formato:

```
-templatePath raíz_instalación/profileTemplates/tipo_plantilla/default
```

donde *tipo_plantilla* es `xs_augment` o `pf_augment`.

Ejemplo

- Utilización de la plantilla xs_augment:

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

- Utilización de la plantilla pf_augment:

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

Crear un perfil de célula

Puede utilizar el mandato manageprofiles para crear un perfil de célula, que está formada por un gestor de despliegue y un servidor de aplicaciones.

Parámetros

Especifique los siguientes parámetros en la plantilla del gestor de despliegue:

-create
Crea un perfil. (Necesario)

-templatePath *vía_acceso_plantilla*
Especifica la vía de acceso de archivo de la plantilla. (Necesario)

Utilice el siguiente formato:

```
-templatePath raíz_instalación/profileTemplates/tipo_plantillatemplate_type/cell/dmgr
```

donde *tipo_plantilla* es `xs_augment` o `pf_augment`.

Especifique los siguientes parámetros con la plantilla del servidor de aplicaciones:

-create
Crea un perfil. (Necesario)

-templatePath *vía_acceso_plantilla*
Especifica la vía de acceso de archivo de la plantilla. (Necesario)

Utilice el siguiente formato:

```
-templatePath raíz_instalación/profileTemplates/tipo_plantilla/cell/default
```

donde *tipo_plantilla* es `xs_augment` o `pf_augment`.

Ejemplo

- Utilización de la plantilla xs_augment:

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

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

- Utilización de la plantilla pf_augment:

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

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

Aumentar un perfil de gestor de despliegue

Puede utilizar el mandato manageprofiles para aumentar un perfil de gestor de despliegue.

Parámetros

-augment
Aumenta el perfil existente. (Necesario)

-profileName
Especifica el nombre del perfil. (Necesario)

-templatePath *vía_acceso_plantilla*
Especifica la vía de acceso de los archivos de plantilla que se encuentran en el directorio raíz de la instalación. (Necesario)

Utilice el siguiente formato:

```
-templatePath raíz_instalación/profileTemplates/tipo_plantilla/dmgr
```

donde *tipo_plantilla* es `xs_augment` o `pf_augment`.

Ejemplo

- Utilización de la plantilla xs_augment:

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

- Utilización de la plantilla pf_augment:

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

Aumentar un perfil personalizado

Puede utilizar el mandato manageprofiles para aumentar un perfil personalizado.

Parámetros

-augment
Aumenta el perfil existente. (Necesario)

-profileName
Especifica el nombre del perfil. (Necesario)

-templatePath *vía_acceso_plantilla*
Especifica la vía de acceso de los archivos de plantilla que se encuentran en el directorio raíz de la instalación. (Necesario)

Utilice el siguiente formato:

```
-templatePath raíz_instalación/profileTemplates/tipo_plantilla/managed
```

donde *tipo_plantilla* es `xs_augment` o `pf_augment`.

Ejemplo

- Utilización de la plantilla xs_augment:

```
./manageprofile.sh|bat -augment -profileName profile01 -templatePath raíz_instalación/profileTemplates/xs_augment/managed
```

- Utilización de la plantilla pf_augment:

```
./manageprofile.sh|bat -augment -profileName profile01 -templatePath raíz_instalación/profileTemplates/pf_augment/managed
```

Aumentar un perfil de servidor de aplicaciones autónomo

Puede utilizar el mandato manageprofiles para aumentar un perfil de servidor de aplicaciones autónomo.

Parámetros

-augment
Aumenta el perfil existente. (Necesario)

-profileName
Especifica el nombre del perfil. (Necesario)

-templatePath *vía_acceso_plantilla*
Especifica la vía de acceso de los archivos de plantilla que se encuentran en el directorio raíz de la instalación. (Necesario)

Utilice el siguiente formato:

```
-templatePath raíz_instalación/profileTemplates/tipo_plantilla/default
```

donde *tipo_plantilla* es `xs_augment` o `pf_augment`.

Ejemplo

- Utilización de la plantilla xs_augment:

```
./manageprofile.sh|bat -augment -profileName profile01 -templatePath raíz_instalación/profileTemplates/xs_augment/default
```

- Utilización de la plantilla pf_augment:

```
./manageprofile.sh|bat -augment -profileName profile01 -templatePath raíz_instalación/profileTemplates/pf_augment/default
```

Aumentar un perfil de célula

Puede utilizar el mandato manageprofiles para aumentar un perfil de célula

Parámetros

Especifique los siguientes parámetros para el perfil de gestor de despliegue:

-augment
Aumenta el perfil existente. (Necesario)

-profileName
Especifica el nombre del perfil. (Necesario)

-templatePath *vía_acceso_plantilla*
Especifica la vía de acceso de los archivos de plantilla que se encuentran en el directorio raíz de la instalación. (Necesario)

Utilice el siguiente formato:

```
-templatePath raíz_instalación/profileTemplates/tipo_plantillatemplate_type/cell/dmgr
```

donde *tipo_plantilla* es `xs_augment` o `pf_augment`.

Especifique los siguientes parámetros para el perfil de servidor de aplicaciones:

-augment
Aumenta el perfil existente. (Necesario)

-profileName
Especifica el nombre del perfil. (Necesario)

-templatePath *vía_acceso_plantilla*
Especifica la vía de acceso de los archivos de plantilla que se encuentran en el directorio raíz de la instalación. (Necesario)

Utilice el siguiente formato:

```
-templatePath raíz_instalación/profileTemplates/tipo_plantilla/cell/default
```

donde *tipo_plantilla* es `xs_augment` o `pf_augment`.

Ejemplo

- Utilización de la plantilla xs_augment:

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

./manageprofile.sh|bat -augment -profileName profile01 -templatePath raíz_instalación/profileTemplates/xs_augment/cell/default
```

- Utilización de la plantilla pf_augment:

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

./manageprofile.sh|bat -augment -profileName profile01 -templatePath raíz_instalación/profileTemplates/pf_augment/cell/default
```

Reducir un perfil

Para reducir un perfil, especifique el parámetro **-ignoreStack** con el parámetro **-templatePath** además de especifica los parámetros **-unaugment** y **-profileName** necesarios.

Parámetros

-unaugment
Reduce un perfil aumentado previamente. (Necesario)

-profileName
Especifica el nombre del perfil. El parámetro se emite de forma predeterminada, si no se especifica ningún valor. (Necesario)

-templatePath *vía_acceso_plantilla*
Especifica la vía de acceso de los archivos de plantilla que se encuentran en el directorio raíz de la instalación. (Opcional)

Utilice el siguiente formato:

donde *tipo_plantilla* es `xs_augment` o `pf_augment` y *tipo_perfil* adopta uno de estos cuatro tipos:

- `dmgr`: perfil de gestor de despliegue
- `managed`: perfil personalizado
- `default`: perfil de servidor de aplicaciones autónomo
- `cell`: perfil de célula

-ignoreStack
Se utiliza con el parámetro **-templatePath** para reducir un perfil determinado que ha sido aumentado. (Opcional)

Ejemplo

- Utilización de la plantilla xs_augment:

```
./manageprofile.sh|bat -unaugment -profileName profile01 -ignoreStack
-templatePath raíz_instalación/profileTemplates/xs_augment/tipo_perfil
```

- Utilización de la plantilla pf_augment:

```
./manageprofile.sh|bat -unaugment -profileName profile01 -ignoreStack
-templatePath raíz_instalación/profileTemplates/pf_augment/tipo_perfil
```

Tareas relacionadas

"Creación y aumento de perfiles para WebSphere eXtreme Scale" en la página 41
Después de instalar el producto, cree tipos exclusivos de perfiles y aumente los existentes para WebSphere eXtreme Scale.

"Utilización de la interfaz gráfica de usuario para crear perfiles" en la página 42
Utilice la interfaz gráfica de usuario (GUI), que proporciona el plug-in de la herramienta de gestión de perfiles para crear perfiles para WebSphere eXtreme Scale. Un perfil es un conjunto de archivos que define el entorno de ejecución.

"Utilización de la interfaz gráfica de usuario para aumentar perfiles" en la página 43
Después de instalar el producto, podrá aumentar un perfil existente para que sea compatible con WebSphere eXtreme Scale.

Perfiles que no son root

Otorgue a un usuario que no es root los permisos para los archivos y directorios de forma que el usuario no root pueda crear un perfil para el producto. El usuario no root también puede aumentar un perfil que fue creado por un usuario root, un usuario no root diferente o el mismo usuario no root.

En general, los usuarios no root están limitados a poder crear y utilizar los perfiles en su entorno. Dentro del plug-in de la herramienta de gestión de perfiles, los nombres y valores de puerto exclusivos están inhabilitaddos para los usuarios no root. El usuario no root debe cambiar los valores de campo predeterminados en la herramienta de gestión de perfiles para el nombre de perfil, nombre de nodo, nombre de célula y asignaciones de puerto. Considere asignar a los usuarios no root un rango de valores para cada uno de los campos. Puede asignar responsabilidad a los usuarios no root para adherir los rangos de valores adecuados y para mantener la integridad de sus propias definiciones.

El término *instalador* hace referencia a un usuario root y, también, a un usuario no root. Como instalador, puede otorgar a los usuarios no root los permisos para crear perfiles y establecer sus propios entornos de producto. Por ejemplo, un usuario no root podría crear un entorno de producto para probar el despliegue de aplicaciones con un perfil que es suyo. Las tareas específicas que puede completar para permitir la creación de un perfil no root incluyen los siguientes elementos:

- La creación de un perfil y la asignación de propiedad del directorio de perfil a un usuario no root, de forma que el usuario no root pueda iniciar WebSphere Application Server para un perfil específico.
- El otorgamiento de permisos de escritura de los archivos y directorios apropiados a un usuario no root, que permite al usuario no root crear el perfil. Con esta tarea, puede crear un grupo para usuarios que están autorizados para crear perfiles, o proporcionar a los usuarios individuales la capacidad de crear perfiles.
- La instalación de los paquetes de mantenimiento para el producto, que incluye los servicios necesarios para los perfiles existentes que son propiedad de un usuario no root. Como instalador, es el propietario de los archivos nuevos que crea el paquete de mantenimiento.

Si desea detalles adicionales, consulte la información detallada sobre cómo crear perfiles para los usuarios no root, que incluye los pasos para completar los ejemplos de la tarea anterior, en el centro de información de WebSphere Application Server Network Deployment.

Como instalador, también puede otorgar permisos para un usuario no root para aumentar perfiles. Por ejemplo, un usuario no root puede aumentar un perfil que ha sido creado por un instalador, o aumentar un perfil que crea éste. Siga el proceso de aumento del usuario no root WebSphere Application Server Network Deployment para completar estas tareas.

Sin embargo, cuando un usuario no root aumenta un perfil que ha sido creado por el instalador, el usuario no tiene que crear los archivos siguientes antes del aumento, porque los archivos se establecieron durante el proceso de creación del perfil:

- *raíz_servidor_aplic*`/logs/manageprofiles.xml`
- *raíz_servidor_aplic*`/properties/fsdb.xml`
- *raíz_servidor_aplic*`/properties/profileRegistry.xml`

Cuando un usuario no root aumenta un perfil que crea, el usuario no root debe modificar los permisos para los documentos que se encuentran dentro de las plantillas del perfil eXtreme Scale.

Instalación de WebSphere eXtreme Scale de forma silenciosa

Utilice un archivo de respuestas plenamente cualificado, que puede configurar de forma específica según sus necesidades, o pasar parámetros a la línea de mandatos para instalar en modalidad silenciosa WebSphere eXtreme Scale .

Antes de empezar

Detenga todos los procesos que se están ejecutando en el entorno WebSphere Application Server o WebSphere Application Server Network Deployment. Consulte Programas de utilidad de línea de mandatos si desea más información sobre los mandatos stopManager, stopNode, y stopServer.

Por qué y cuándo se efectúa esta tarea

Una instalación silenciosa utiliza el mismo programa de instalación que utiliza la versión de la interfaz gráfica de usuario (GUI). Sin embargo, en lugar de visualizar una interfaz de asistente, la instalación silenciosa lee todas las respuestas de un archivo que personaliza, o de los parámetros que pase a la línea de mandatos.

Consulte wxssetup.response.txt para ver un archivo de respuestas de ejemplo y una descripción de cada opción.

1. Opcional: Si opta por instalar eXtreme Scale utilizando un archivo de respuestas, en primer lugar, personalícelo.

   Nota: Debe especificar el nombre de archivo de respuestas completo. La especificación de la vía de acceso relativa provoca que la instalación falles sin ninguna indicación de que se haya producido un error.

   a. Copie el archivo de respuestas del DVD del producto en la unidad de disco.
   b. Abra y edite el archivo de respuestas en el editor de texto que elija. Debe especificar los siguientes parámetros:
      - El acuerdo de licencia
      - El directorio de instalación

      Consejo: Cuando instale eXtreme Scale en un entorno WebSphere Application Server, el instalador utiliza el directorio de instalación para determinar dónde está instalada la instancia del WebSphere Application Server existente. Si realiza la instalación en un nodo que contiene varias instancias de WebSphere Application Server, defina claramente la ubicación.
   c. Ejecute el siguiente script para iniciar la instalación.

```
./install.sh|bat -options C:/vía_acceso_unidad/archivo_respuestas.txt -silent
```

2. Opcional: Si elige instalar eXtreme Scale pasando determinados parámetros a la línea de mandatos, ejecute el siguiente script para iniciar la instalación:

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

Parámetros de instalación

Especifique los parámetros en la línea de mandatos para personalizar y configurar la instalación del producto.

Nota: Debe especificar el nombre de archivo de respuestas completo. La especificación de la vía de acceso relativa provoca que la instalación falles sin ninguna indicación de que se haya producido un error.

Parámetros

Puede pasar los siguientes parámetros durante una instalación del producto a través de la línea de mandatos o de un archivo de opciones:

-silent
Elimina la interfaz gráfica de usuario (GUI). Especifique el parámetro **-options** para indicar que el instalador completa la instalación de acuerdo con un archivo de opciones personalizadas. Si no especifica el parámetro **-options**, en su lugar se utilizan los valores predeterminados.

Ejemplo de uso

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

-options *nombre_vía/nombre_archivo*
Especifica un archivo de opciones que utiliza el instalador para completar una instalación silenciosa. Las propiedades de la línea de mandatos tienen preferencia.

Ejemplo de uso

```
./install.sh|bat -options c:/nombre_vía/archivo_opciones.txt
```

-log # !nombre_archivo *@tipo_suceso*

Genera un archivo de registro de instalación que anota los siguientes tipos de sucesos:

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

Ejemplo de uso

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

-is:log *nombre_vía/nombre_archivo*

Crea un archivo de registro que contiene las búsquedas de la máquina virtual Java (JVM) del instalador mientras intenta iniciar la GUI. El archivo de registro no se crea, a menos que se especifique.

Ejemplo de uso

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

-is:javaconsole

Visualiza una ventana de consola durante el proceso de instalación.

Ejemplo de uso

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

-is:silent

Elimina la ventana de inicialización de Java que, normalmente, se visualiza cuando se inicia el instalador.

Ejemplo de uso

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

-is:tempdir *nombre_vía*

Especifica el directorio temporal que utiliza el instalador durante la instalación.

Ejemplo de uso

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

Tareas relacionadas

"Instalación de WebSphere eXtreme Scale autónomo" en la página 28
Puede instalar el WebSphere eXtreme Scale autónomo en un entorno que no contiene WebSphere Application Server o WebSphere Application Server Network Deployment.

"Desinstalación de WebSphere eXtreme Scale" en la página 56
Para eliminar WebSphere eXtreme Scale del entorno, puede utilizar el asistente o puede desinstalar de forma silenciosa el producto.

"Integración de WebSphere eXtreme Scale con WebSphere Application Server" en la página 31
Puede instalar WebSphere eXtreme Scale en un entorno en el que WebSphere Application Server o WebSphere Application Server Network Deployment está instalado. Puede utilizar las características existentes de WebSphere Application Server o WebSphere Application Server Network Deployment para mejorar las aplicaciones aplicando la capacidad de eXtreme Scale.

"Creación y aumento de perfiles para WebSphere eXtreme Scale" en la página 41
Después de instalar el producto, cree tipos exclusivos de perfiles y aumente los existentes para WebSphere eXtreme Scale.

Utilización del instalador de actualización para instalar los paquetes de mantenimiento

Utilice IBM Update Installer para actualizar el entorno de WebSphere eXtreme Scale con distintos tipos de mantenimiento como, por ejemplo, arreglos temporales, fixpacks y paquetes de renovación.

Por qué y cuándo se efectúa esta tarea

Utilice el instalador de actualización de IBM para instalar y aplicar distintos tipos de paquetes de mantenimiento para WebSphere eXtreme Scale. Puesto que el instalador de actualización realiza mantenimientos regulares, debe utilizar la versión más actual de la herramienta.

1. Detenga todos los procesos que se están ejecutando en el entorno.
   - Para detener todos los procesos que se están ejecutando en el entorno eXtreme Scale autónomo, consulte "Detención de servidores de eXtreme Scale autónomos" en la página 239 si desea más información.
   - Para detener todos los procesos que se están ejecutando en el entorno WebSphere Application Server, consulte Programas de utilidad de línea de mandatos.
2. Descargue la versión más reciente del instalador de actualización. Consulte los Arreglos recomendados si desea más información.
3. Instale el instalador de actualización. Consulte Instalación del instalador de actualización para el software WebSphere en el centro de información de WebSphere Application Server si desea más información.
4. Descargue los paquetes de mantenimiento en el directorio *raíz_updi*/`maintenance` en el que tiene previsto realizar la instalación. Consulte el Sitio de soporte si desea más información.
5. Utilice el instalador de actualización para instalar el arreglo temporal, el fixpack o el paquete de renovación. Puede instalar el paquete de mantenimiento ejecutando la interfaz gráfica de usuario (GUI), o ejecutando el instalador de actualización en la modalidad silenciosa.

   Ejecute el siguiente mandato desde el directorio *raíz_updi* para iniciar la GUI:

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

Ejecute el siguiente mandato desde el directorio *raíz_updi* para ejecutar el instalador de actualización en la modalidad silenciosa:

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

Si falla el proceso de instalación, consulte el archivo de registro temporal, que está en el directorio *raíz_updi*`/logs/update/tmp`. El instalador de actualización crea el directorio *raíz_instalación*`/logs/update/`*paquete_mantenimiento*`.install` en el que se encuentran los archivos de registro de la instalación.

Configuración de un intermediario de solicitud de objetos personalizado

Puede utilizar una versión personalizada del intermediario de solicitud de objetos (ORB) con WebSphere eXtreme Scale cuando ejecute procesos Java Platform, Standard Edition autónomos en el entorno.

Antes de empezar

WebSphere eXtreme Scale y WebSphere Application Server proporcionan ambos un ORB, que ya está configurado para ser utilizado con eXtreme Scale. Bajo circunstancias normales, no tiene que configurar el ORB ni utilizar un ORB diferente.

Por qué y cuándo se efectúa esta tarea

WebSphere eXtreme Scale utiliza el intermediario de solicitud de objetos (ORB) para habilitar la comunicación entre los procesos. Un ORB se incluye con eXtreme Scale y WebSphere Application Server. Si utiliza un IBM Developer Kit o SDK proporcionado con WebSphere Application Server, el ORB se incluye en el JRE.

Puede utilizar el ORB proporcionado con eXtreme Scale, el ORB proporcionado con IBM SDK, o el ORB que se suministra con WebSphere Application Server. Cualquier problema que encuentre al utilizar los ORB de proveedores de software independientes debe ser reproducible con el ORB de IBM y compatible con JRE antes de ponerse en contacto con el equipo de soporte. eXtreme Scale no soporta el ORB que se suministra con Sun Microsystems Java Development Kit (JDK). Mientras que eXtreme Scale soporta los kits de desarrollador de la mayoría de los proveedores, se recomienda que utilice el ORB proporcionado con eXtreme Scale.

- Si el entorno contiene una versión 5 SDK o posterior, actualice los scripts que inician el mandato Java especificando un directorio alternativo.
  1. Copie el archivo `ibmorb.jar` personalizado y el archivo `ibmorbapi.jar` en un directorio vacío.
  - Complete el paso siguiente cuando utilice los scripts del producto en un entorno eXtreme Scale autónomo:
    1. Edite la vía de acceso de la variable OBJECTGRID_ENDORSED_DIRS en el archivo setupCmdLine para hacer referencia al directorio ORB personalizado. Guarde los cambios.

Edite el archivo *objectgridRoot*/bin/setupCmdLine.sh.

Edite el archivo *objectgridRoot*\bin\setupCmdLine.bat.

– Complete el paso siguiente cuando utilice los scripts de producto en un entorno WebSphere Application Server:
  1. Añada los siguientes parámetros y propiedad del sistema en el script startOgServer:

     -jvmArgs –Djava.endorsed.dirs=*directorio_ORB_personalizado*

– Complete el paso siguiente cuando utilice un script personalizado para iniciar un proceso de aplicación cliente o un proceso de servidor:
  1. Añada la siguiente propiedad del sistema al script personalizado:

     -Djava.endorsed.dirs=*directorio_ORB_personalizado*

- Si el entorno contiene una versión 1.4.2 SDK, integre el ORB de IBM en el SDK especificado.
  1. Descargue y extraiga el ORB de un IBM SDK. Si no está disponible IBM SDK para su plataforma, descargue y extraiga el IBM Developer Kit de Linux, Java Technology Edition. Consulte IBM developer kits si desea más información.
  2. Copie los archivos java/jre/lib/ibmorb.jar y java/jre/lib/ibmorbapi.jar en el directorio java/jre/lib/ext en el SDK de destino.
  3. Cree o edite el archivo orb.properties, que está en el directorio java/jre/lib de SDK. Añada las siguientes propiedades o verifique que las siguientes propiedades existen en el archivo:

     ```
     org.omg.CORBA.ORBClass=com.ibm.CORBA.iiop.ORB
     org.omg.CORBA.ORBSingletonClass=com.ibm.rmi.corba.ORBSingleton
     org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.objectgrid.corba.ObjectGridInitializer
     ```

     Si desea descripciones de las propiedades y los valores, consulte “Archivo de propiedades ORB” en la página 207.
  4. Descargue Xerces2 Java 2.9. Consulte The Apache Xerces Project - Descargas si desea más información.
  5. Copie los archivos xercesImpl.jar y xml-apis.jar en el directorio lib/ext.

Resultados

Puede utilizar el ORB personalizado con eXtreme Scale cuando ejecute los procesos cliente y servidor de Java SE autónomo.

Referencia relacionada

“Archivo de propiedades ORB” en la página 207
El archivo orb.properties se utiliza para pasar las propiedades utilizadas por el intermediario de solicitud de objetos (ORB) para modificar el comportamiento de transporte de la cuadrícula.

Propiedades personalizadas del intermediario para solicitudes de objetos

Desinstalación de WebSphere eXtreme Scale

Para eliminar WebSphere eXtreme Scale del entorno, puede utilizar el asistente o puede desinstalar de forma silenciosa el producto.

Antes de empezar

Atención: El desinstalador elimina todos los archivos binarios y todo el mantenimiento como, por ejemplo, fix packs y arreglos temporales, a la vez.

1. Detenga todos los procesos que están ejecutando eXtreme Scale. De lo contrario, la desinstalación fallará.
   - Si ha instalado el eXtreme Scale autónomo, consulte “Detención de servidores de eXtreme Scale autónomos” en la página 239.
   - Si ha instalado eXtreme Scale con una instalación existente de WebSphere Application Server, consulte Programas de utilidad de línea de mandatos si desea más información sobre cómo detener procesos WebSphere Application Server.
2. Ejecute el siguiente script:
   - UNIX Linux `raíz_instalación/uninstall_wxs/uninstall.`
   - Windows `raíz_instalación\uninstall_wxs\uninstall.exe`

Resultados

Ha eliminado eXtreme Scale del entorno.

Referencia relacionada

“Parámetros de instalación” en la página 52
Especifique los parámetros en la línea de mandatos para personalizar y configurar la instalación del producto.

Capítulo 5. Personalización de WebSphere eXtreme Scale para z/OS

Mediante el uso de WebSphere Customization Tools, puede generar y ejecutar trabajos personalizados para personalizar WebSphere eXtreme Scale para z/OS.

Antes de empezar

- Verifique que el sistema contiene el nivel más reciente de WebSphere Application Server Network Deployment:
  - Si está ejecutando la versión 6.1, el sistema debe contener un Fixpack 27, como mínimo. Consulte Instalación del entorno de servicio de aplicaciones de la versión 6.1 si desea más información.
  - Si está ejecutando la versión 7.0, el sistema debe contener el Fixpack 3, como mínimo. Consulte Instalación del entorno de servicio de aplicaciones de la versión 7.0 si desea más información.
- Instale WebSphere eXtreme Scale para z/OS. Consulte el directorio del programa WebSphere eXtreme Scale en la Página de la biblioteca si desea más información.

Por qué y cuándo se efectúa esta tarea

Mediante el uso de WebSphere Customization Tools, genere definiciones de personalización y suba y ejecute los trabajos personalizados para personalizar WebSphere eXtreme Scale para z/OS. Consulte los temas siguientes si desea más información:

- "Instalación de WebSphere Customization Tools"
- "Generación de definiciones de personalización" en la página 60
- "Subida y ejecución de trabajos personalizados" en la página 61

Instalación de WebSphere Customization Tools

Instale WebSphere Customization Tools versión 7.0.0.3 o posterior para personalizar el entorno de WebSphere eXtreme Scale para z/OS.

Antes de empezar

Instale WebSphere eXtreme Scale para z/OS. Consulte el directorio del programa WebSphere eXtreme Scale en la Página de la biblioteca si desea más información.

Por qué y cuándo se efectúa esta tarea

WebSphere Customization Tools es una herramienta gráfica basada en una estación de trabajo que se utiliza para crear trabajos de personalización que generan entornos de tiempo de ejecución de WebSphere eXtreme Scale para z/OS.

1. Utilice el FTP para copiar los archivos de ampliación `xs.wct` y `xspf.wct` del sistema z/OS a la estación de trabajo en la que está instalando WebSphere Customization Tools. Los archivos de ampliación se encuentran en el directorio `/usr/lpp/zWebSphereXS/util/V7R0/WCT` del sistema z/OS.
2. Descargue e instale WebSphere Customization Tools versión 7.0.0.3 o posterior desde el sitio web apropiado:

- Windows WebSphere Customization Tools for Windows
- Linux WebSphere Customization Tools for Linux

3. Suba el archivo `xs.wct` a la aplicación de WebSphere Customization Tools.
   a. Inicie la aplicación WebSphere Customization Tools en la estación de trabajo.
   b. Pulse **Ayuda → Actualizaciones de software → Instalar ampliación**.
   c. Desde el panel Ubicaciones de la ampliación de WebSphere Customization Tools, pulse **Instalar nueva ubicación de ampliación**.
   d. Desde el panel Archivo de archivado de origen, pulse **Examinar**, vaya hasta al directorio en el que ha copiado el archivo `xs.wct` en el paso 1 y pulse **Abrir**.
   e. Pulse **Siguiente** en el panel Resumen.

      Nota: Se visualiza el panel Instalación correcta. Antes de poder pulsar **Finalizar**, debe copiar y guardar los datos del campo de ubicación:

      ```
      C:\Documents and Settings\Administrator\WCT\workspace\configuration\
      com.ibm.ws.pmt.update\com.ibm.ws390.pmt.xs_7.0.0.0\eclipse
      ```

   f. Desde el panel Configuración del producto, pulse **Añadir una ubicación de ampliación**. Pegue los datos que ha copiado en el paso anterior en el campo Ubicación y pulse **Aceptar**.
   g. Pulse **Sí** para reiniciar WebSphere Customization Tools.
4. Suba el archivo `xspf.wct` a la aplicación WebSphere Customization Tools.
   a. Pulse **Ayuda → Actualizaciones de software → Instalar ampliación**.
   b. Desde el panel Ubicaciones de la ampliación de WebSphere Customization Tools, pulse **Instalar nueva ubicación de ampliación**.
   c. Desde el panel Archivo de archivado de origen, pulse **Examinar**, vaya hasta el directorio en el que ha copiado el archivo `xspf.wct` en el paso 1 y pulse **Abrir**.
   d. Pulse **Siguiente** en el panel Resumen.

      Nota: Se visualiza el panel Instalación correcta. Antes de poder pulsar **Finalizar**, debe copiar y guardar los datos del campo de ubicación:

      ```
      C:\Documents and Settings\Administrator\WCT\workspace\configuration\
      com.ibm.ws.pmt.update\com.ibm.ws390.pmt.xs_7.0.0.0\eclipse
      ```

   e. Desde el panel Configuración del producto, pulse **Añadir una ubicación de ampliación**. Pegue los datos que ha copiado en el paso anterior en el campo Ubicación y pulse **Aceptar**.
   f. Pulse **Sí** para reiniciar WebSphere Customization Tools.

Qué hacer a continuación

Después de subir los archivos de ampliación y reiniciar WebSphere Customization Tools, puede utilizar la Herramienta de gestión de perfiles para generar definiciones personalizados para eXtreme Scale para z/OS. Si desea más información, consulte “Generación de definiciones de personalización”.

Generación de definiciones de personalización

Utilice la función de Herramienta de gestión de perfiles dentro de WebSphere Customization Tools para generar definiciones de personalización y crear trabajos personalizados para WebSphere eXtreme Scale para z/OS.

Antes de empezar

Instale WebSphere Customization Tools y suba los archivos de ampliación `xs.wct` y `xspf.wct`. Si desea más información, consulte “Instalación de WebSphere Customization Tools” en la página 59.

Por qué y cuándo se efectúa esta tarea

Puede generar definiciones de personalización utilizando la Herramienta de gestión de perfiles, que se proporciona en WebSphere Customization Tools. Una *definición de personalización* es un conjunto de archivos utilizados para crear trabajos personalizados con la finalidad de configurar WebSphere eXtreme Scale para z/OS.

1. Inicie la Herramienta de gestión de perfiles.
   - Windows Pulse **Inicio** → **Programas** → **IBM WebSphere** → **WebSphere Customization Tools**. Después de que se inicie la aplicación, pulse la pestaña **Herramienta de gestión de perfiles**.
   - Linux Pulse *menús_sistema_operativa* → **IBM WebSphere** → **WebSphere Customization Tools**. Después de que se inicie la aplicación, pulse la pestaña **Herramienta de gestión de perfiles**.
2. Añada una ubicación existente o cree una nueva ubicación de la definición de personalización que desee crear. En la pestaña **Ubicaciones de personalización**, pulse **Añadir**. Si crea una nueva ubicación, el recuadro Versión hace referencia a la versión del producto WebSphere Application Server existente instalada en el sistema z/OS.

   Nota: No utilice la misma ubicación que utiliza para otras definiciones de personalización de eXtreme Scale.
3. Genere la definición de personalización. En la pestaña **Definiciones de personalización**, pulse **Aumentar**.
4. Seleccione el tipo de entorno de definición para crear:
   - Nodo de servidor de aplicaciones autónomo
   - Gestor de despliegue
   - Servidor de aplicaciones
   - Nodo gestionado (personalizado)
5. Complete los campos de los paneles. Especifique los valores para los parámetros que se han utilizado para crear el sistema z/OS.
6. Pulse **Aumentar** para generar la definición de personalización.

Qué hacer a continuación

Suba el trabajo personalizado en el sistema z/OS de destino. Si desea más información, consulte “Subida y ejecución de trabajos personalizados”.

Subida y ejecución de trabajos personalizados

Después de generar las definiciones de personalización, puede subir y ejecutar los trabajos personalizados asociados a las definiciones en el sistema WebSphere eXtreme Scale para z/OS.

Antes de empezar

Genere las definiciones de personalización para los trabajos que desea subir al sistema z/OS. Si desea más información, consulte “Generación de definiciones de personalización” en la página 60.

Por qué y cuándo se efectúa esta tarea

Suba y ejecute los trabajos personalizados que ha creado mediante WebSphere Customization Tools para administrar y supervisar WebSphere eXtreme Scale para el entorno de z/OS.

1. Suba los trabajos personalizados. En la pestaña **Definiciones de personalización**, seleccione los trabajos que desea subir y pulse **Procesar**.
2. Suba los trabajos al servidor FTP en el sistema z/OS. Especifique la información necesaria en el panel **Subir definición de personalización**.
3. Pulse **Finalizar**.
4. Ejecute los trabajos personalizados. Pulse la pestaña **Instrucciones de personalización** y siga las instrucciones de personalización para cada trabajo.

Capítulo 6. Configuración de WebSphere eXtreme Scale

Puede configurar WebSphere eXtreme Scale para ejecutarse en un entorno autónomo, o puede configurar eXtreme Scale para ejecutarse en un entorno que contiene WebSphere Application Server o WebSphere Application Server Network Deployment. Para que un despliegue de eXtreme Scale adopte los cambios de configuración en la cuadrícula del servidor, debe reiniciar los procesos para que estos cambios entren en vigor, en lugar de aplicarlos de forma dinámica. Sin embargo, en el cliente, aunque no puede alterar los valores de configuración para una instancia de cliente existente, puede crear un nuevo cliente con los valores que necesite utilizando un archivo XML o mediante programas. Al crear un cliente, puede alterar temporalmente los valores predeterminados que proceden de la configuración de servidor actual.

Configuración de un ObjectGrid local en memoria

Se puede crear una configuración de eXtreme Scale en memoria local mediante el uso de un archivo XML de descriptor de ObjectGrid o de las API de eXtreme Scale.

Por qué y cuándo se efectúa esta tarea

A continuación un ejemplo sencillo de XML: el archivo `companyGrid.xml`. Las primeras líneas del archivo incluyen la cabecera necesaria para cada archivo XML de ObjectGrid. El archivo define el ObjectGrid CompanyGrid con las BackingMaps Customer, Item, OrderLine y Order. Un archivo XML de política de despliegue se pasa a un contenedor eXtreme Scale durante el arranque.

Archivo 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>
```

Pase el archivo XML a uno de los métodos createObjectGrid en la interfaz ObjectGridManager. El siguiente código de ejemplo valida el archivo `companyGrid.xml` respecto al esquema XML y crea el ObjectGrid CompanyGrid. La instancia de ObjectGrid recién creada no se almacena en memoria caché.

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

Como alternativa al uso de XML, los objetos ObjectGrid se pueden crear a través de programa. El siguiente ejemplo de código se puede utilizar en lugar del código y XML anterior.

```
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");
```

eXtreme Scale tiene muchos plug-ins y atributos personalizables. Si desea una descripción completa del archivo XML de ObjectGrid, consulte la eXtreme Scale referencia de la configuración.

Interfaz ObjectGrid

Los siguientes métodos le permiten interactuar con una instancia de ObjectGrid.

Introducción

Crear e inicializar

Consulte el tema sobre la interfaz ObjectGridManager para obtener información sobre los pasos necesarios para crear una instancia ObjectGrid. Existen dos métodos distintos para crear una instancia ObjectGrid: mediante programa o mediante configuración de los archivos XML. Consulte la documentación de la API si desea más información.

Get o set y métodos de fábrica

Antes de inicializar la instancia ObjectGrid, debe llamarse a los métodos set. Si llama a un método set después de que se llame al método initialize, se genera una excepción java.lang.IllegalStateException. Cada uno de los métodos getSession de la interfaz ObjectGrid llaman implícitamente al método initialize. Por lo tanto, debe llamar a los métodos set antes de llamar a los métodos getSession. La única excepción a esta regla se produce al establecer, añadir y eliminar los objetos ObjectGridEventListener. Estos objetos pueden procesarse después de terminado el proceso de inicialización.

La interfaz ObjectGrid contiene los siguientes métodos principales.

Tabla 5. Métodos de interfaz ObjectGrid

Método	Descripción
BackingMap defineMap(String name);	defineMap: es un método de fábrica para definir un BackingMap de nombre exclusivo. Para obtener más información sobre las correlaciones de respaldo, consulte la interfaz BackingMap.
BackingMap getMap(String name);	getMap: devuelve un objeto BackingMap previamente definido al llamar a defineMap. Al usar este método, puede configurar BackingMap, si no se ha configurado ya a través de la configuración del XML.
BackingMap createMap(String name);	createMap: crea un objeto BackingMap, pero no lo almacena en caché para que lo use este ObjectGrid. Utilice este método con el método setMaps(List) de la interfaz ObjectGrid, que almacena en la memoria caché BackingMaps para ser utilizados con este ObjectGrid. Utilice estos métodos al configurar un objeto ObjectGrid con la infraestructura Spring.
void setMaps(List mapList);	setMaps: borra los objetos BackingMap que se hayan definido previamente en este ObjectGrid y los sustituye por la lista de BackingMaps proporcionados.
public Session getSession() throws ObjectGridException, TransactionCallbackException;	getSession: devuelve un objeto Session, que proporciona funcionalidad de inicio, confirmación y retrotracción para una unidad de trabajo. Para obtener más información sobre los objetos Session, consulte la interfaz Session.
Session getSession(CredentialGenerator cg);	getSession(CredentialGenerator cg): obtiene una sesión con un objeto CredentialGenerator. Sólo puede llamar a este método un cliente ObjectGrid en un entorno cliente/servidor.
Session getSession(Subject subject);	getSession(Subject subject): permite el uso de un objeto Subject específico en lugar del configurado en ObjectGrid para obtener un objeto Session.

Tabla 5. Métodos de interfaz ObjectGrid (continuación)

Método	Descripción
void initialize() throws ObjectGridException;	initialize: se inicializa ObjectGrid y está disponible para su uso general. Este método se llama de forma implícita cuando se llama al método getSession, si ObjectGrid no está en una fase inicializada.
void destroy();	destroy: la infraestructura se desensambla y no se puede utilizar después de llamar a este método.
void setTxTimeout(int timeout);	setTxTimeout: utilice este método para establecer la cantidad de tiempo, en segundos, que tardará en completarse una transacción iniciada por una sesión que creó esta instancia ObjectGrid. Si una transacción no se completa en el tiempo especificado, el objeto Session que inició la transacción se marca como "tiempo de espera excedido". Marcar un objeto Session como con el tiempo de espera excedido provoca que el siguiente método ObjectMap invocado por el Session de tiempo de espera genere una excepción . El objeto se marca sólo como retrotracción, lo cual provoca que la transacción se retrotraiga aunque la aplicación llame al método commit en lugar de al método rollback después de que la excepción TransactionTimeoutException haya sido captada por la aplicación. Un valor de tiempo de espera de 0 indica que a la transacción se le permite un tiempo ilimitado para completarse. Si se utiliza el valor de 0, no se producirá tiempo de espera excedido por parte de la transacción. Si no se llama a este método, cualquier objeto Session devuelto por el método getSession de esta interfaz tiene un valor predeterminado de tiempo de espera de la transacción establecido en 0. Una aplicación puede alterar temporalmente el valor de tiempo de espera de la transacción por objeto Session mediante el uso del método setTransactionTimeout de la interfaz com.ibm.websphere.objectgrid.Session. También puede configurar el tiempo de espera de transacción con el archivo `objectGrid.xml` en el caso distribuido.
int getTxTimeout();	getTxTimeout: devuelve el valor del tiempo de espera de la transacción en segundos. Este método devuelve el mismo valor pasado como parámetro de tiempo de espera en el método setTxTimeout. Si no se llama al método setTxTimeout, el método devuelve 0 para indicar que a la transacción se le permite un tiempo ilimitado para completarse.
public int getObjectGridType();	Devuelve el tipo de ObjectGrid. El valor devuelto equivale a una de las constantes declaradas en este interfaz: LOCAL, SERVER o CLIENT.
public void registerEntities(Class[] entities);	Registra una o más entidades basadas en los metadatos de la clase. El registro de entidades debe realizarse antes de la inicialización de ObjectGrid para enlazar un objeto Entity con BackingMap y cualquier índice definido. Este método puede llamarse varias veces.
public void registerEntities(URL entityXML);	Registra una o más entidades de un archivo XML de entidad. El registro de entidades debe realizarse antes de la inicialización de ObjectGrid para enlazar un objeto Entity con BackingMap y cualquier índice definido. Este método puede llamarse varias veces.
void setQueryConfig(QueryConfig queryConfig);	Establece el objeto QueryConfig para este ObjectGrid. Un objeto QueryConfig proporciona configuraciones de consulta para ejecutar consultas de objetos en las correlaciones de este ObjectGrid.
//Palabras clave	
void associateKeyword(Serializable parent, Serializable child);	En desuso: utilice la función de índice o consultas para obtener objetos con atributos específicos. El método associateKeyword proporciona un mecanismo de invalidación flexible basado en palabras clave. Este método enlaza las dos palabras clave en una relación direccional. Si el padre se invalida, el hijo también. La invalidación del hijo no afecta al padre.
//Seguridad	
void setSecurityEnabled()	setSecurityEnabled: habilita la seguridad. La seguridad está inhabilitada de manera predeterminada.
void setPermissionCheckPeriod(long period);	setPermissionCheckPeriod: este método acepta un solo parámetro que indica la frecuencia con la que debe comprobarse el permiso utilizado para permitir el acceso de un cliente. Si el parámetro es 0, todos los métodos solicitan el mecanismo de autorización, ya sea autorización JAAS o autorización personalizada, para comprobar si el sujeto actual tiene permiso. Puede que esta estrategia provoque problemas de rendimiento en función de la implementación de la autorización. No obstante, este tipo de autorización está disponible si se prefiere. De forma alternativa, si el parámetro es menor que 0, indica el número de milisegundos para almacenar en memoria caché un conjunto de permisos antes de volver al mecanismo de autorización para que los actualice. Con este parámetro mejora el rendimiento, pero si se modifican los permisos del programa de fondo durante este tiempo, ObjectGrid puede permitir o prohibir el acceso aunque se haya modificado el proveedor de seguridad de programa de fondo.
void setAuthorizationMechanism(int authMechanism);	setAuthorizationMechanism: establece el mecanismo de autorización. El valor predeterminado es SecurityConstants.JAAS_AUTHORIZATION.

Tabla 5. Métodos de interfaz ObjectGrid (continuación)

Método	Descripción
setMapAuthorization(MapAuthorization ma);	En desuso: utilice el método setObjectGridAuthorization (ObjectGridAuthorization) en lugar de conectar las autorizaciones personalizadas. setMapAuthorization: establece el plug-in MapAuthorization para esta instancia ObjectGrid. Este plug-in se puede utilizar para autorizar los accesos de ObjectMap o JavaMap a los principales contenidos en el objeto Subject. Una implementación típica de este plug-in es recuperar los principales del objeto Subject y después comprobar si se han concedido los permisos especificados a los principales.
setSubjectSource(SubjectSource ss);	setSubjectSource: establece el plug-in SubjectSource. Este plug-in puede utilizarse para obtener un objeto Subject que represente el cliente ObjectGrid. Este objeto Subject se utiliza para la autorización de ObjectGrid. El tiempo de ejecución de ObjectGrid llama al método SubjectSource.getSubject cuando se utiliza el método ObjectGrid.getSession para obtener una sesión y se ha habilitado la seguridad. Este plug-in resulta útil para un cliente ya autenticado: puede recuperar un objeto Subject autenticado y pasar después a la instancia ObjectGrid. No se necesita otra autenticación.
setSubjectValidation(SubjectValidation sv);	setSubjectValidation: establece el plug-in SubjectValidation para esta instancia ObjectGrid. Este plug-in puede utilizarse para validar que un sujeto javax.security.auth.Subject pasado a ObjectGrid es un sujeto válido que no se ha manipulado de forma indebida. Una implementación de este plug-in necesita el soporte del creador del objeto Subject porque sólo el creador sabe si el objeto Subject ha sido manipulado indebidamente. Puede darse el caso, no obstante, de que el creador no sepa si el objeto Subject se ha manipulado de forma indebida. De ser así, este plug-in no debería utilizarse.
void setObjectGridAuthorization(ObjectGrid Authorization ogAuthorization);	Establece ObjectGridAuthorization para esta instancia ObjectGrid. Si se pasan valores nulos a este método, se elimina un objeto ObjectGridAuthorization ya establecido de una invocación anterior de este método, e indica que este <code>ObjectGrid</code> no está asociado a ningún objeto ObjectGridAuthorization. Este método sólo debe utilizarse cuando se ha habilitado la seguridad ObjectGrid. Si la seguridad ObjectGrid está inhabilitada, no se utilizará el objeto ObjectGridAuthorization proporcionado. Puede utilizarse un plug-in ObjectGridAuthorization para autorizar el acceso a ObjectGrid y las correlaciones. Desde XD 6.1, setMapAuthorization está en desuso y se recomienda el uso de setObjectGridAuthorization. No obstante, si se utilizan los plug-in MapAuthorization y ObjectGridAuthorization, ObjectGrid utilizará el plug-in MapAuthorization proporcionado para autorizar los accesos de la correlaciones, aunque esté en desuso.

Interfaz ObjectGrid: plug-ins

La interfaz ObjectGrid tiene varios puntos de plug-in opcionales para interacciones más ampliables.

```
void addEventListener(ObjectGridEventListener cb);
void setEventListeners(List cbList);
void removeEventListener(ObjectGridEventListener cb);
void setTransactionCallback(TransactionCallback callback);
int reserveSlot(String);
// Plug-ins relacionados con la seguridad
void setSubjectValidation(SubjectValidation subjectValidation);
void setSubjectSource(SubjectSource source);
void setMapAuthorization(MapAuthorization mapAuthorization);
```

- ObjectGridEventListener: se utiliza una interfaz ObjectGridEventListener para recibir notificaciones cuando se producen sucesos significativos en ObjectGrid. Estos sucesos pueden ser inicialización de ObjectGrid, inicio de una transacción, fin de una transacción y destrucción de ObjectGrid. Para escuchar estos sucesos, cree una clase que implemente la interfaz ObjectGridEventListener y la añada a ObjectGrid. Estas escuchas se asocian con cada objeto Session. Consulte el tema que trata sobre escuchas y la interfaz Session para obtener más información.
- TransactionCallback: una interfaz de escucha TransactionCallback permite que se envíen sucesos transaccionales como señales de inicio, confirmación y retrotracción a esta interfaz. Por norma, se suele utilizar una interfaz de escucha TransactionCallback con un cargador. Para obtener más información, consulte el tema que trata sobre el plug-in TransactionCallback y los cargadores. Estos sucesos pueden utilizarse para coordinar transacciones con un recurso externo o dentro de varios cargadores.

- reserveSlot: permite que los plug-ins de este ObjectGrid reserven ranuras para su uso en instancias de objetos que tienen ranuras como TxID.
- SubjectValidation: si se habilita la seguridad, este plug-in puede utilizarse para validar una clase javax.security.auth.Subject que se pasa a ObjectGrid.
- MapAuthorization: si se habilita la seguridad, este plug-in puede utilizarse para autorizar los accesos de ObjectMap a los principales que se representan en el objeto Subject.
- SubjectSource: si se habilita la seguridad, este plug-in puede utilizarse para obtener un objeto Subject que represente el cliente ObjectGrid. A continuación, este objeto Subject se utiliza para la autorización de ObjectGrid.

Si desea más información sobre los plug-ins, consulte la introducción de los plug-ins en *Guía de programación*.

Interfaz BackingMap

Cada instancia de ObjectGrid contiene una colección de objetos BackingMap. Utilice el método defineMap o el método createMap de la interfaz ObjectGrid para nombrar y añadir cada BackingMap a una instancia de ObjectGrid. Estos métodos devuelven una instancia BackingMap que se utiliza después para definir el comportamiento de una correlación individual.

Interfaz Session

La interfaz Session se utiliza para iniciar una transacción y obtener el objeto ObjectMap o JavaMap necesario para realizar la interacción transaccional entre una aplicación y un objeto BackingMap. No obstante, los cambios de la transacción no se aplican al objeto BackingMap hasta que se confirme la transacción. Un objeto BackingMap puede considerarse como una memoria caché en memoria de datos confirmados para una correlación individual. Si desea más información, consulte la información sobre cómo utilizar Sessions para acceder a los datos de *Guía de programación*.

La interfaz BackingMap proporciona métodos para establecer los atributos BackingMap. Algunos de estos métodos permiten la ampliación de un objeto BackingMap a través de diversos plug-ins de diseño personalizado. Consulte la lista siguiente de los métodos set para definir atributos y proporcionar soporte de plug-ins de diseño personalizado:

```
// Para establecer atributos de
BackingMap.
public void setReadOnly(boolean readOnlyEnabled);
public void setNullValuesSupported(boolean nullValuesSupported);
public void setLockStrategy( LockStrategy lockStrategy );
public void setCopyMode(CopyMode mode, Class valueInterface);
public void setCopyKey(boolean b);
public void setNumberOfBuckets(int numBuckets);
public void setNumberOfLockBuckets(int numBuckets);
public void setLockTimeout(int seconds);
public void setTimeToLive(int seconds);
public void setTtlEvictorType(TTLType type);
public void setEvictionTriggers(String evictionTriggers);

// Para establecer un plug-in personalizado y opcional que proporciona la
aplicación.
public abstract void setObjectTransformer(ObjectTransformer t);
public abstract void setOptimisticCallback(OptimisticCallback checker);
public abstract void setLoader(Loader loader);
public abstract void setPreloadMode(boolean async);
public abstract void setEvictor(Evictor e);
public void setMapEventListeners( List /*MapEventListener*/ eventListenerList );
public void addMapEventListener(MapEventListener eventListener );
public void removeMapEventListener(MapEventListener eventListener );
public void addMapIndexPlugin(MapIndexPlugin index);
public void setMapIndexPlugins(List /\* MapIndexPlugin \*/ indexList );
```

```
public void createDynamicIndex(String name, boolean isRangeIndex,
String attributeName, DynamicIndexCallback cb);
public void createDynamicIndex(MapIndexPlugin index, DynamicIndexCallback cb);
public void removeDynamicIndex(String name);
```

Para cada uno de los métodos set listados, existe un método get correspondiente.

Atributos de BackingMap

Cada BackingMap tiene los siguientes atributos que pueden establecerse para modificar o controlar el comportamiento del objeto BackingMap:

- ReadOnly: este atributo indica si la correlación es una correlación de sólo lectura o una correlación de lectura y grabación. Si este atributo no se establece, la correlación toma el valor predeterminado de lectura y grabación. Cuando un objeto BackingMap se establece para sólo lectura, ObjectGrid optimiza el rendimiento para sólo lectura siempre que sea posible.
- NullValuesSupported: este atributo indica si se puede asignar un valor nulo a la correlación. Si no se establece este atributo, la correlación no admite valores nulos. Si la correlación admite valores nulos, una operación get que devuelve un valor nulo significa que o bien el valor es nulo o bien la correlación no contiene la clave especificada por la operación get.
- LockStrategy: este atributo determina si este objeto BackingMap utiliza un gestor de bloqueos. Si se utiliza, el atributo LockStrategy se usa para indicar si se emplea un procedimiento de bloqueo optimista o pesimista para bloquear las entradas de correlación. Si no se establece este atributo, se utiliza la estrategia de bloqueo optimista. Consulte el tema sobre bloqueos para obtener más información sobre las estrategias de bloqueo permitidas.
- CopyMode: este atributo determina si BackingMap realiza una copia de un objeto de valor cuando se lee un valor de la correlación o éste se coloca en el objeto BackingMap durante el ciclo de confirmación de una transacción. Se admiten diversas modalidades de copia que permiten a la aplicación conseguir el equilibrio entre rendimiento e integridad de datos. Si no se establece el atributo, se utiliza la modalidad de copia COPY_ON_READ_AND_COMMIT. Esta modalidad de copia no tiene el mejor rendimiento, pero sí la mejor protección frente a problemas de integridad de los datos. Si BackingMap está asociado a una entidad de API EntityManager, el valor de CopyMode no tiene ningún efecto, a menos que el valor se establezca en COPY_TO_BYTES. Si se define cualquier otro CopyMode, siempre estará establecido en NO_COPY. Para alterar temporalmente el valor de CopyMode para una correlación de entidad, utilice el método ObjectMap.setCopyMode. Si desea más información sobre las modalidades de copia, consulte la información sobre los procedimientos recomendados del método en *Guía de programación*.
- CopyKey: este atributo determina si BackingMap realiza una copia de un objeto de clave cuando se crea por primera vez una entrada en la correlación. De manera predeterminada, no se realiza ninguna copia de objetos de clave porque las claves suelen ser objetos que no se pueden cambiar.
- NumberOfBuckets: este atributo indica el número de grupos hash que BackingMap utiliza. La implementación de BackingMap utiliza una correlación hash para su implementación. Si existen muchas entradas en el objeto BackingMap, la existencia de más grupos mejora el rendimiento. El número de claves que tienen el mismo grupo disminuye a medida que crece el número de grupos. Un número mayor de grupos también implica mayor simultaneidad. Este atributo es útil para el ajuste del rendimiento. Se utiliza un valor predeterminado no definido si la aplicación no establece el atributo NumberOfBuckets.

- NumberOfLockBuckets: este atributo indica el número de grupos de bloqueo que utiliza el gestor de bloqueos para este objeto BackingMap. Cuando LockStrategy se establece en OPTIMISTIC o PESSIMISTIC, se crea un gestor de bloqueos para BackingMap. El gestor de bloqueos utiliza una correlación hash para realizar un seguimiento de las entradas bloqueadas por una o más transacciones. Si existen muchas entradas en la correlación hash, un número mayor de grupos de bloqueo mejora el rendimiento porque el número de claves que tienen el mismo grupo disminuye a medida que crece el número de grupos. Un número mayor de grupos de bloqueo también implica mayor simultaneidad. Si el atributo LockStrategy se establece en NONE, este objeto BackingMap no utiliza ningún gestor de bloqueos. En este caso, establecer numberOfLockBuckets no tiene ningún efecto. Si no se establece este atributo, se utiliza un valor no definido.
- LockTimeout: este atributo se utiliza cuando BackingMap usa un gestor de bloqueos. BackingMap utiliza un gestor de bloqueos cuando el atributo LockStrategy está establecido en OPTIMISTIC o PESSIMISTIC. El valor del atributo se especifica en segundos y determina cuánto tiempo espera el gestor de bloqueos a que se otorgue un bloqueo. Si este atributo no está definido, se utilizan 15 segundos como el valor de LockTimeout. Consulte el tema que trata sobre el bloqueo pesimista para obtener más información sobre las excepciones de tiempo de espera de bloqueo que pueden producirse.
- TtlEvictorType: cada BackingMap tiene su propio desalojador de tiempo de vida incorporado que utiliza un algoritmo basado en la hora para determinar las entradas de la correlación que debe desalojar. De manera predeterminada, el desalojador de tiempo de vida incorporado no está activo. Para activarlo, llame al método setTtlEvictorType con uno de los tres valores: CREATION_TIME, LAST_ACCESS_TIME o NONE. Un valor CREATION_TIME indica que el desalojador añade el atributo TimeToLive a la hora en que se creó la entrada de correlación en BackingMap para determinar cuándo debe el desalojador desalojar la entrada de correlación del objeto BackingMap. Un valor LAST_ACCESS_TIME indica que el desalojador añade el atributo TimeToLive a la hora en la que una transacción, que ejecuta la aplicación, accedió por última vez a la entrada de correlación para determinar cuándo debe el desalojador desalojar la entrada de correlación. La entrada de correlación sólo se desaloja si ninguna transacción accede a una entrada de correlación durante un período de tiempo especificado por el atributo TimeToLive. Un valor NONE indica que el desalojador debe permanecer inactivo y no desalojar nunca ninguna de las entradas de correlación. Si no se establece este atributo, NONE se utiliza como el valor predeterminado y el desalojador de tiempo de vida no está activo. Consulte el tema sobre los desalojadores para obtener más información sobre el desalojador de tiempo de vida incorporado.
- TimeToLive: este atributo se utiliza para especificar el número de segundos que necesita el desalojador de tiempo de vida incorporado para añadir al tiempo de creación o de acceso por última vez de cada entrada, como se ha descrito en el atributo TtlEvictorType. Si no se establece este atributo, se utiliza el valor especial de cero para indicar que el tiempo de vida es infinito. Si este atributo se establece en infinito, el desalojador nunca desaloja las entradas de correlación.

El ejemplo siguiente muestra cómo definir someMap BackingMap en la instancia someGrid ObjectGrid y establecer varios atributos de BackingMap mediante los métodos set de la interfaz BackingMap:

```
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("someGrid");
BackingMap bm = objectGrid.getMap("someMap");
bm.setReadOnly( true ); // alterar temporalmente el valor predeterminado de
                         // lectura/grabación
bm.setNullValuesSupported(false); // alterar temporalmente el valor
                        // predeterminado de permitir valores nulos
bm.setLockStrategy( LockStrategy.PESSIMISTIC ); // alterar temporalmente
                        // el valor predeterminado de OPTIMISTIC
bm.setLockTimeout( 60 ); // alterar temporalmente el valor predeterminado
                        // de 15 segundos.
bm.setNumberOfBuckets(251); // alterar temporalmente el valor predeterminado
                        // (números primos funcionan mejor)
bm.setNumberOfLockBuckets(251); // alterar temporalmente el valor
                        // predeterminado (números primos funcionan mejor)
```

Plug-ins de BackingMap

La interfaz BackingMap tiene diversos plug-ins opcionales que amplían las interacciones con ObjectGrid:

- **Plug-in ObjectTransformer**: para algunas operaciones de correlación, BackingMap podría necesitar serializar, deserializar o copiar una clave o valor de una entrada en el objeto BackingMap. BackingMap puede realizar estas acciones; para ello, proporcione una implementación predeterminada de la interfaz ObjectTransformer. Una aplicación puede mejorar el rendimiento si proporciona un plug-in ObjectTransformer de diseño personalizado que BackingMap utilice para serializar, deserializar o copiar una clave o valor de una entrada en BackingMap. Consulte la información sobre el plug-in ObjectTransformer en *Guía de programación* si desea más información.
- **Plug-in Evictor**: el desalojador de tiempo de vida incorporado utiliza un algoritmo basado en la hora para decidir cuándo se debe desalojador una entrada en BackingMap. Algunas aplicaciones podrían necesitar un algoritmo diferente para decidir cuándo debe desalojarse una entrada en BackingMap. El plug-in Evictor pone a disposición de BackingMap un desalojador de diseño personalizado. El plug-in Evictor se ofrece además del desalojador de tiempo de vida incorporado. No so reemplaza. ObjectGrid proporciona un plug-in Evictor personalizado que implementa los algoritmos conocidos de ″menos utilizado recientemente″ o ″utilizado con menor frecuencia″. Las aplicaciones pueden conectar uno de los plug-ins Evictor suministrados o pueden proporcionar su propio plug-in Evictor. Consulte la información sobre el desalojo en *Guía de programación*.
- **Plug-in MapEventListener**: es posible que una aplicación desee conocer los sucesos de BackingMap como, por ejemplo, un desalojo de una entrada de correlación o una precarga de una terminación de BackingMap. BackingMap llama a los métodos del plug-in MapEventListener para informar de sucesos de BackingMap a una aplicación. Una aplicación puede recibir una notificación de varios sucesos de BackingMap mediante el método setMapEventListener para proporcionar uno o más plug-ins MapEventListener de diseño personalizado a BackingMap. La aplicación puede modificar los objetos MapEventListener listados utilizando el método addMapEventListener o el método removeMapEventListener. Consulte el la información sobre el plug-in MapEventListener en *Guía de programación* si desea más información.
- **Plug-in Loader**: BackingMap es una memoria caché en memoria de una correlación. Un plug-in es una opción que utiliza BackingMap para mover datos entre memorias y se utiliza para un almacén persistente de BackingMap. Por ejemplo, se puede utilizar un cargador JDBC (Java database connectivity) para insertar y extraer datos de BackingMap y una o más tablas relacionales de una base de datos relacional- Una base de datos relacional no necesita utilizarse como almacén persistente de un objeto BackingMap. El cargador también se puede utilizar para mover datos entre BackingMap y un archivo, entre

BackingMap y una correlación Hibernate, entre BackingMap y un bean de entidad de Java 2 Platform, Enterprise Edition (JEE), entre BackingMap y otro servidor de aplicaciones, etc. La aplicación debe proporcionar un plug-in Loader de diseño personalizado para mover datos entre BackingMap y el almacén persistente de cada tecnología que se use. Si no se proporciona un cargador, BackingMap se convierte en una sencilla memoria caché en memoria. Consulte la información sobre cómo utilizar un cargador en *Guía de programación* si desea más información.

- **Plug-in OptimisticCallback**: cuando el atributo LockStrategy para BackingMap está establecido en OPTIMISTIC, BackingMap o un plug-in Loader debe realizar operaciones de comparación para los valores de la correlación. BackingMap y Loader utilizan el plug-in OptimisticCallback para realizar las operaciones de comparación de versiones optimistas. Consulte la información sobre el plug-in OptimisticCallback en *Guía de programación* si desea más información.
- **Plug-in MapIndexPlugin**: un plug-in MapIndexPlugin, o un Index, es una opción utilizada por BackingMap para crear un índice que se basa en el atributo especificado del objeto almacenado. El índice permite a la aplicación buscar objetos por un valor específico o un intervalo de valores. Existen dos tipos de índice: estático y dinámico. Consulte el tema que trata sobre los índices para obtener más información.

Si desea más información en relación a los plug-ins, consulte la introducción a los plug-ins en *Guía de programación*.

Bloqueo de entrada de correlación

ObjectGrid BackingMap admite diversas estrategias de bloqueo para mantener la coherencia de las entradas en memoria caché.

Cada BackingMap puede configurarse de modo que utilice una de las estrategias de bloqueo siguientes:

1. Modalidad de bloqueo optimista
2. Modalidad de bloqueo pesimista
3. Ninguno

La estrategia de bloqueo predeterminada es OPTIMISTIC. Utilice el bloqueo optimista cuando los datos no se modifican frecuentemente. Los bloqueos sólo se mantienen durante un tiempo breve mientras los datos se leen de la memoria caché y se copian en la transacción. Cuando la memoria caché de la transacción se sincroniza con la memoria caché principal, los objetos de la memoria caché actualizados se comprueban contra la versión original. Si la comprobación falla, la transacción se retrotrae y se produce la excepción OptimisticCollisionException.

La estrategia de bloqueo PESSIMISTIC adquiere bloqueos para las entradas de memoria caché y debe utilizarse cuando los datos se cambian con frecuencia. Cada vez que se lee una entrada de la memoria caché, se adquiere un bloqueo, que puede mantenerse condicionalmente hasta que se complete la transacción. La duración de algunos de los bloqueos pueden ajustarse mediante el uso de niveles de aislamiento para la sesión.

Si el bloqueo no es necesario porque los datos nunca se actualizan o sólo se actualizan durante períodos tranquilos, puede inhabilitar el bloqueo mediante el uso de la estrategia de bloqueo NONE. Esta estrategia es muy rápida porque no se necesita ningún gestor de bloqueos. La estrategia de bloqueo NONE es ideal en tablas de búsqueda o en correlaciones de sólo lectura.

Si desea más información sobre las estrategias de bloqueo, consulte la información sobre las estrategias de bloqueo en *Visión general del producto*.

Especificación de una estrategia de bloqueo

El siguiente ejemplo demuestra cómo se puede establecer la estrategia de bloqueo en las correlaciones BackingMaps map1, map2 y map3, donde cada correlación utiliza una estrategia de bloqueo diferente. El primer fragmento de código muestra cómo utilizar el XML para la configuración de la estrategia de bloqueo y el segundo fragmento de código muestra un enfoque mediante programa.

Enfoque de XML

Configuración de BackingMap - XML de ejemplo
```
<?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="map1"
                lockStrategy="PESSIMISTIC" numberOfLockBuckets="31"/>
            <backingMap name="map2"
                lockStrategy="OPTIMISTIC" numberOfLockBuckets="409"/>
            <backingMap name="map3"
                lockStrategy="NONE"/>
        </objectGrid>
    </objectGrids>
</objectGridConfig>
```

Enfoque mediante programa

Configuración de BackingMap - ejemplo mediante programa
```
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("map1");
bm.setLockStrategy( LockStrategy.PESSIMISTIC );
bm.setNumberOfLockBuckets(31);
bm = og.defineMap("map2");
bm.setNumberOfLockBuckets(409);
bm.setLockStrategy( LockStrategy.OPTIMISTIC );
bm = og.defineMap("map3");
bm.setLockStrategy( LockStrategy.NONE );
```

Para evitar una excepción de java.lang.IllegalStateException, se debe llamar al método setLockStrategy antes de utilizar los métodos initialize o getSession en una instancia local de ObjectGrid.

Configuración del gestor de bloqueos

Cuando se utiliza una estrategia de bloqueo PESSIMISTIC u OPTIMISTIC, se crea un gestor de bloqueos para BackingMap. El gestor de bloqueos utiliza una correlación hash para realizar un seguimiento de las entradas bloqueadas por una o más transacciones. Cuantas más entradas de correlación existan en la correlación hash, mayor será el grupo de bloqueos con un buen rendimiento. El riesgo de las colisiones de sincronización de Java es menor a medida que crece el número de grupos. Un número mayor de grupos también implica mayor simultaneidad. El ejemplo anterior muestra cómo una aplicación puede establecer el número de grupos de bloqueos para utilizar en una instancia determinada de BackingMap.

Para evitar una excepción java.lang.IllegalStateException, debe llamarse al método setNumberOfLockBuckets antes que a los métodos initialize o getSession en la instancia ObjectGrid. El parámetro del método setNumberOfLockBuckets es un entero primitivo de Java que especifica el número de grupos de bloqueo para

utilizar. El uso de un número primo puede permitir una distribución uniforme de entradas de correlación en los grupos de bloqueos. Un buen punto de partida para obtener un mejor rendimiento es establecer el número de grupos de bloqueos en un diez por ciento del número esperado de entradas de BackingMap.

Configuración de una estrategia de bloqueo

Puede definir una estrategia de bloqueo optimista, pesimista o sin bloqueo en cada BackingMap en la configuración de WebSphere eXtreme Scale.

Por qué y cuándo se efectúa esta tarea

Puede especificar una estrategia de bloqueo a través de programa o con XML. Si desea más información sobre el bloqueo, consulte la información sobre las estrategias de bloque en *Visión general del producto*.

- **Configure una estrategia de bloqueo optimista**
  - A través de programas

```
especifique la estrategia optimista a través de programasimport 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 );
```

  - Mediante XML

```
especifique la estrategia optimista mediante 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="test">
            <backingMap name="optimisticMap"
                lockStrategy="OPTIMISTIC"/>
        </objectGrid>
    </objectGrids>
</objectGridConfig>
```

- **Configure una estrategia de bloqueo pesimista**
  - A través de programas

```
especifique la estrategia pesimista a través de programaimport 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);
```

  - Mediante XML

```
especifique la estrategia pesimista mediante 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="test">
            <backingMap name="pessimisticMap"
                lockStrategy="PESSIMISTIC"/>
        </objectGrid>
    </objectGrids>
</objectGridConfig>
```

- **Configure una estrategia sin bloqueo**
  - A través de programas

```
especifique una estrategia sin bloqueo a través de programas
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);
```

– Mediante XML

```
especifique una estrategia sin bloqueo con 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="test">
            <backingMap name="noLockingMap"
                lockStrategy="NONE"/>
        </objectGrid>
    </objectGrids>
</objectGridConfig>
```

Qué hacer a continuación

Para evitar que se genere una excepción java.lang.IllegalStateException, debe llamar al método setLockStrategy antes de llamar a los métodos initialize o getSession en la instancia de ObjectGrid.

Configuración de una cuadrícula distribuida de eXtreme Scale

Utilice el archivo XML de descriptor de la política de despliegue para gestionar la topología de despliegue.

La política de despliegue está codificada como un archivo XML que se proporciona para un contenedor eXtreme Scale. El archivo XML especifica la siguiente información:

- Las correlaciones que pertenecen a cada conjunto de correlaciones
- El número de particiones
- El número de réplicas síncronas y asíncronas

Si desea más información sobre cómo iniciar los servidores de contenedor, consulte “Inicio de procesos de contenedor en un entorno deWebSphere Application Server” en la página 244 o “Inicio de procesos de contenedor” en la página 232.

La política de despliegue también controla los siguientes comportamientos de colocación.

- El número mínimo de contenedores activos antes de que se lleve a cabo la colocación
- Sustitución automática de fragmentos perdidos
- Colocación de cada fragmento desde una sola partición en otra máquina

Si desea más información sobre el archivo XML de la política de despliegue, consulte “Archivo XML de descriptor de la política de despliegue” en la página 152.

La información de punto final no está preconfigurada en el entorno dinámico. En la política de despliegue no hay ningún nombre de servidor ni información de topología física. Todos los fragmentos de una cuadrícula se colocan automáticamente en contenedores a través del servicio de catálogo. El servicio de catálogo utiliza las restricciones definidas por la política de despliegue para

gestionar automáticamente la colocación de fragmentos. Esta colocación de fragmentos automática lleva una fácil configuración para las cuadrículas grandes. También puede añadir servidores al entorno según sea necesario.

Nota: En un entorno WebSphere Application Server, no está soportado un tamaño de grupo principal de más de 50 miembros.

Un archivo XML de política de despliegue se pasa a un contenedor eXtreme Scale durante el arranque. Se debe utilizar una política de despliegue junto con un archivo XML de ObjectGrid. La política de despliegue no es necesaria para iniciar un contenedor, aunque se recomienda. La política de despliegue debe ser compatible con el XML de ObjectGrid que se utiliza con la misma. Para cada elemento objectgridDeployment de la política de despliegue, debe tener un objectGrid correspondiente en el XML de ObjectGrid. Las correlaciones en objectgridDeployment deben ser coherentes con las backingMaps encontradas en el XML de ObjectGrid. Debe hacerse referencia a cada backingMap dentro de únicamente un mapSet.

En el siguiente ejemplo, se intenta emparejar el archivo `companyGridDpReplication.xml` con el correspondiente archivo `companyGrid.xml`.

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>
```

El archivo `companyGridDpReplication.xml` tiene un mapSet dividido en 11 particiones. Cada partición debe tener, exactamente, una réplica síncrona. El número de réplicas síncronas viene dictado por los atributos minSyncReplicas y maxSyncReplicas. Puesto que el atributo minSyncReplicas está establecido en 1, cada partición de mapSet debe tener, como mínimo, una réplica síncrona disponible para procesar transacciones de escritura. Puesto que maxSyncReplicas está establecido en 1, cada partición no debe superar una réplica síncrona. Las particiones de este mapSet no tiene réplicas asíncronas.

El atributo numInitialContainers instruye al servicio de catálogo que difiera la colocación hasta que estén disponibles cuatro contenedores para soportar este ObjectGrid. El atributo numInitialContainers se ignora después de que se haya alcanzado el número especificado de contenedores.

El archivo `companyGridDpReplication.xml` muestra una manera habitual de configurar una política de despliegue, aunque una política de despliegue puede ofrecer incluso más control sobre cómo y cuándo se despliega ObjectGrid en el entorno. Si desea una descripción completa del archivo descriptor de la política de despliegue, consulte "Archivo XML de descriptor de la política de despliegue" en la página 152.

Topología distribuida

Las memorias caché coherentes distribuidas ofrecen un mayor rendimiento, disponibilidad y escalabilidad que puede configurar el usuario.

WebSphere eXtreme Scale equilibra automáticamente los servidores. Puede incluir servidores adicionales sin reiniciar WebSphere eXtreme Scale. La adición de servidores adicionales sin tener que reiniciar eXtreme Scale le permite tener despliegues sencillos y, también, despliegues grandes de terabytes en los que son necesarios cientos de servidores. Esta topología de despliegue es flexible. Con el servicio de catálogo, puede añadir y eliminar servidores para utilizar mejor los recursos sin eliminar toda la memoria caché. Para añadir o eliminar un servidor, utilice el mandato startOgServer para iniciar un servidor de contenedor, que se comunica con el servicio de catálogo con la opción **-catalogServiceEndPoints**. Todos los clientes de la topología distribuida se comunican con el servicio de catálogo a través del protocolo IIOP (Internet Interoperability Object Protocol). Todos los clientes utilizan la interfaz de ObjectGrid para comunicarse con los servidores.

La prestación de la configuración dinámica de WebSphere eXtreme Scale facilita la adición de recursos al sistema. Los contenedores alojan los datos y las funciones de servicio de catálogo como punto táctil para la cuadrícula. Los contenedores son responsables del mantenimiento de datos. El servicio de catálogo es responsable del reenvío de solicitudes al lugar correcto la primera vez, asignando espacio de contenedores de host, y gestionando el estado y la disponibilidad del sistema en general. Los clientes se conectan a un servicio de catálogo, recuperan una descripción de la topología de contenedor-servidor, y luego se comunican directamente con cada servidor cuando sea necesario. Cuando la topología del servidor cambia debido a la adición de nuevos servidores, o debido a la anomalía de otros, el se direcciona automáticamente al servidor apropiado que aloja los datos.

Normalmente, un servicio de catálogo existe en su propia cuadrícula de Máquinas virtuales Java . Se puede utilizar un único servicio de catálogo para gestionar diversos servidores. Un contenedor se puede iniciar en una JVM por sí mismo o puede cargar en una JVM arbitraria con otros contenedores para distintos servidores. Un cliente puede existir en cualquier JVM y comunicarse con uno o más servidores. También puede existir un cliente en la misma JVM que la de un contenedor.

También puede crear una política de despliegue a través de programas al incorporar un contenedor en un proceso o aplicación Java existente. Si desea más información, consulte la API de eXtreme Scale DeploymentPolicy.

Configuración de un cliente de eXtreme Scale

Puede configurar un cliente de eXtreme Scale basándose en los requisitos, incluidos los valores de alteración temporal.

Puede configurar un cliente eXtreme Scale de las formas siguientes:

- Configuración a través de XML
- Configuración mediante programa
- Configuración de la infraestructura Spring
- Inhabilitación de la memoria caché cercana

Puede alterar temporalmente los siguientes plug-ins en un cliente.

- **Plug-ins ObjectGrid**
  - Plug-in TransactionCallback
  - Plug-in ObjectGridEventListener
- **Plug-ins de BackingMap**
  - Plug-in Evictor
  - Plug-in MapEventListener
  - Atributo numberOfBuckets
  - Atributo ttlEvictorType
  - Atributo timeToLive

Configuración del cliente con XML

Un archivo XML ObjectGrid se puede utilizar para alterar los valores en el lado del cliente. Para cambiar los valores en un cliente de eXtreme Scale, debe crear un archivo XML de ObjectGrid que sea similar en estructura al archivo que se utilizó para el servidor eXtreme Scale.

Presuponga que el siguiente archivo XML se emparejó con un archivo XML de política de despliegue, y que estos archivos se utilizaron para iniciar un servidor eXtreme Scale.

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>
```

En un servidor eXtreme Scale, CompanyGrid se comporta según se haya definido en el archivo `companyGridServerSide.xml`. De forma predeterminada, el cliente de CompanyGrid tiene los mismos valores que los de CompanyGrid que se ejecuta en el servidor. No obstante, algunos de los valores se pueden alterar temporalmente en el cliente.

Cree un ObjectGrid específico del cliente para alterar temporalmente algunos de estos valores. Copie el archivo XML de ObjectGrid que se utilizó para abrir el servidor y edite el archivo para personalizar el lado del cliente. Para eliminar un plug-in del cliente, utilice la serie vacía como el valor para el atributo className. Para cambiar un plug-in existente, especifique un nuevo valor para el atributo className. Un plug-in también se puede añadir al archivo si es uno de los plug-ins de alteración temporalmente soportados.

Para establecer uno de los atributos en el cliente, especifique un nuevo valor.

El siguiente archivo XML de ObjectGrid se puede utilizar para especificar algunos de los atributos y plug-ins en el cliente de CompanyGrid.

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>
```

El archivo `companyGridClientSide.xml` altera temporalmente varios atributos y plug-ins en el cliente CompanyGrid. En el cliente, TransactionCallback será com.company.MyClientTxCallback al contrario que com.company.MyTxCallback, que se ejecuta en el servidor. ObjectGridEventListener se elimina del cliente porque el valor de className es la serie vacía.

La backingMap Customer tiene sus numberOfBuckets establecidos en 1429 en el cliente. Customer backingMap conserva Evictor en la configuración de servidor, pero MapEventListener se elimina del cliente.

`numberOfBuckets` y `timeToLive` se han ajustado en el cliente de backingMap OrderLine.

El atributo lockStrategy en este archivo se ignora independientemente del valor que se haya especificado. Este atributo no puede alterarse temporalmente en el cliente.

Para crear el cliente de CompanyGrid utilizando el archivo `companyGridClientSide.xml`, pase el archivo XML de ObjectGrid como un URL a uno de los métodos connect en ObjectGridManager.

Creación del cliente para XML

```
ObjectGridManager ogManager =
 ObjectGridManagerFactory.ObjectGridManager();
 ClientClusterContext clientClusterContext =
 ogManager.connect("MyServer1.company.com:2809", null, new URL(
                "file:xml/companyGridClientSide.xml"));
```

Configuración del cliente mediante programa

También puede alterar temporalmente los valores de ObjectGrid del lado del cliente mediante programa. Cree un objeto ObjectGridConfiguration que sea similar en estructura al ObjectGrid del lado del servidor. El siguiente código construirá un ObjectGrid del lado del cliente que es funcionalmente equivalente a lo que se hubiera creado mediante el companyGridClientSide.xml que se proporcionó en la sección anterior.

alteración temporal del lado del cliente mediante programa

```
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);
```

Sólo los objetos ObjectGridConfiguration y BackingMapConfiguration que se han incluido en overrideMap se comprobarán para los plug-ins y los atributos alterados temporalmente. Para la instancia, el código anterior alterará temporalmente el número de cubetas en la correlación OrderLine. Sin embargo, la correlación Order permanecerá sin modificar en el lado del cliente porque no se ha incluido ninguna configuración.

Configuración del cliente en la infraestructura Spring

Los valores de ObjectGrid del lado del cliente también se pueden alterar temporalmente utilizando la infraestructura Spring. El siguiente archivo XML de ejemplo muestra cómo crear un ObjectGridConfiguration, y utilizarlo para alterar temporalmente algunos valores del lado del cliente. Este ejemplo llama a las mismas API que se han demostrado en la configuración mediante programa. El ejemplo también es funcionalmente equivalente al ejemplo de la configuración del XML de ObjectGrid.

```
configuración de cliente con 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>
```

Tras crear el archivo XML, cargue el archivo y cree el ObjectGrid con el siguiente fragmento de código.

```
BeanFactory beanFactory = new XmlBeanFactory(new
  UrlResource("file:test/companyGridSpring.xml"));

ObjectGrid companyGrid = (ObjectGrid) beanFactory.getBean("companyGrid");
```

Si desea más información, consulte “Integración con la infraestructura Spring” en la página 210.

Inhabilitación de la memoria caché cercana

La memoria caché cercana se habilita de manera predeterminada al configurar el bloqueo como optimista o ninguno, y no puede utilizarse si se configura como pesimista.

Para inhabilitar la memoria caché cercana, debe establecer el atributo de numberOfBuckets en 0 en el archivo descriptor de ObjectGrid de alteración temporal del cliente.

Consulte la información sobre el bloqueo de la entrada de correlación en *Guía de administración* si desea más información.

Habilitación del mecanismo de invalidación de clientes

En un entorno de WebSphere eXtreme Scale distribuido, el cliente tiene una memoria caché cercana de manera predeterminada al utilizar la estrategia de bloqueo optimista o al inhabilitar el bloqueo. La memoria caché cercana tiene sus propios datos locales almacenados en memoria caché. Si un cliente de eXtreme Scale confirma una actualización, la actualización se produce en el servidor y la memoria caché cercana del cliente. Sin embargo, otros clientes de eXtreme Scale no reciben la información de actualización y podrían tener datos desfasados.

Memoria caché cercana

Las aplicaciones deben estar al tanto del problema de los datos obsoletos en el cliente de eXtreme Scale. Puede utilizar la clase ObjectGridEventListener incorporada basada en el Java Message Service (JMS), com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener, para habilitar el mecanismo de invalidación de cliente dentro de un entorno eXtreme Scale distribuido que es conocido como una cuadrícula de eXtreme Scale.

El mecanismo de invalidación de clientes es la solución al problema de los datos obsoletos de la memoria caché cercana del cliente en el entorno distribuido de

eXtreme Scale. Este mecanismo garantiza que la memoria caché cercana del cliente se sincronice con los servidores u otros clientes. Sin embargo, a pesar de este mecanismo de invalidación de clientes basado en JMS, la memoria caché cercana del cliente no se actualiza inmediatamente. Se produce un retardo cuando el tiempo de ejecución de eXtreme Scale publica actualizaciones.

Están disponibles dos modelos para el mecanismo de invalidación de cliente en un entorno de eXtreme Scale distribuido:

- Modelo cliente/servidor: en este modelo, todos los procesos de servidor desempeñan el rol de editor que publica todos los cambios de las transacciones en el destino JMS designado. Todos los procesos de cliente desempeñan los roles de receptor y reciben todos los cambios transaccionales del destino JMS designado.
- Cliente como modelo de roles duales: en este modelo, los procesos de servidor no tienen nada que ver con el destino JMS. Todos los procesos de cliente desempeñan los roles tanto de editor JMS como de receptor. Los cambios transaccionales que se producen en el cliente se publican en el destino JMS y todos los clientes reciben estos cambios transaccionales.

Si desea más información sobre cómo habilitar el mecanismo de invalidación del cliente, consulte “Receptor de sucesos JMS” en la página 146.

Modelo cliente/servidor

En un modelo cliente/servidor, los servidores desempeñan un rol de editor JMS y el cliente desempeña un rol de receptor JMS.

ejemplo de XML del modelo cliente-servidor

```
<?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>
```

Cliente como modelo de roles duales

En un modelo de cliente como roles duales, cada cliente desempeña los roles tanto de editor JMS como de receptor. El cliente publica cada cambio transaccional confirmado en un destino JMS designado y recibe todos los cambios transaccionales confirmados de otros clientes. En este modelo el servidor no tiene nada que ver con JMS.

ejemplo de XML de modelo de roles duales

```
<?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>
```

Configuración del tiempo de espera de reintento de solicitud

Con correlaciones fiables, puede proporcionar un tiempo de espera de reintento a WebSphere eXtreme Scale. El tiempo de espera se utiliza con el tiempo de espera de la transacción para volver a intentar las solicitudes de transacción.

Existen dos métodos para configurar correlaciones fiables. Si el valor está establecido en un valor mayor que cero, la solicitud se intenta hasta que se cumple la condición de tiempo de espera o hasta que se produce una anomalía permanente como, por ejemplo, se ha encontrado una excepción DuplicateKeyException. Si el valor se establece en cero, se utiliza la modalidad fail-fast y eXtreme Scale no realiza más intentos.

Proporcione un valor de tiempo de espera en milisegundos en el archivo de propiedades de cliente o en la sesión. La sesión siempre altera temporalmente el valor de las propiedades de cliente. Durante el tiempo de ejecución, se utiliza el tiempo de espera de transacción con el tiempo de espera de reintento, asegurándose de que el tiempo de espera de reintento no excede el tiempo de espera de transacción.

Debido a variaciones en la forma como se completan las transacciones de compromiso automático y de no compromiso automático (las transacciones que utilizan explícitamente los métodos begin y commit), las excepciones válidas para los reintentos son diferentes.

Para las transacciones que son llamadas dentro de una sesión, el reintento es válido para las excepciones CORBA SystemExceptions y eXtreme Scale TargetNotAvailable.

Para las transacciones de compromiso automático, el reintento es válido para las excepciones de disponibilidad CORBA SystemExceptions eXtreme Scale (ReplicationVotedToRollbackTransactionException, TargetNotAvailable, AvailabilityException, y otras).

Si desea más información, consulte el tema sobre el uso de las sesiones para acceder a los datos de la cuadrícula en *Guía de programación*.

Las anomalías de aplicación u otras anomalías permanentes se devuelven de forma inmediata y no se vuelve a intentar realizar la transacción. Estas anomalías permanentes incluyen las excepciones DuplicateKeyException y KeyNotFoundException.

El valor fail-fast devuelve todas las excepciones sin reintentar ninguna excepción.

La siguiente lista muestra las excepciones de forma más detallada:

Excepciones en las que el cliente realiza reintentos

- ReplicationVotedToRollbackTransactionException (sólo en compromiso automático)
- TargetNotAvailable
- org.omg.CORBA.SystemException
- AvailabilityException (sólo en compromiso automático)
- LockTimeoutException (sólo en compromiso automático)
- UnavailableServiceException (sólo en compromiso automático)

Otras excepciones

- DuplicateKeyException
- KeyNotFoundException
- LoaderException
- TransactionAffinityException
- LockDeadlockException
- OptimisticCollisionException

Definición de la propiedad requestRetryTimeout en un archivo de propiedades de cliente

Para establecer el valor requestRetryTimeout en un cliente, añadir o modificar la propiedad requestRetryTimeout en "Archivo de propiedades de cliente" en la página 203. Las propiedades de cliente están en el archivo `objectGridClient.properties` de forma predeterminada. La propiedad requestRetryTimeout se establece en milisegundos. Establézcalo en un valor mayor que cero para la solicitud que se debe reintentar en las excepciones para las que está disponible el reintento. Establezca el valor en `0` para fail sin reintentos en las excepciones. Para utilizar el comportamiento predeterminado, elimine la propiedad o establezca el valor en `-1`.

objectGridClient.properties

```
# eXtreme Scale client config
preferLocalProcess = false
preferLocalhost = false
requestRetryTimeout = 30000
```

El valor requestRetryTimeout se especifica en milisegundos. En el ejemplo anterior, si el valor se utiliza en una instancia de ObjectGrid, el valor requestRetryTimeout es 30000 milisegundos o 30 segundos.

Definición de las propiedades de cliente obteniendo la conexión de ObjectGrid a través de programa

Para establecer las propiedades de cliente a través de un programa, en primer lugar, cree un archivo de propiedades. En el siguiente ejemplo, el archivo de propiedades de cliente es el archivo `objectGridClient.properties`. Después de que se establezca una conexión con ObjectGridManager, establezca las propiedades de cliente. A continuación, obtenga una instancia de ObjectGrid. Las propiedades de cliente se establecen en dicha instancia de ObjectGrid. Siempre que se modifican las propiedades de cliente, se obtiene una nueva instancia de ObjectGrid.

```
ObjectGridManager manager = ObjectGridManager.instance();
String objectGridName = "testObjectGrid";
URL overRideObjectGridXml = null;
ClientClusterContext ccc = manager.connect("localhost:2809", null, overRideObjectGridXml);
File file= new File("test/objectGridClient.properties");
URL url=file.toURL();
ccc.setClientProperties(objectGridName, url);
ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid objectGrid = ogManager.getObjectGrid(ctx, objectGridName);
```

Ejemplo de modificación temporal de sesión con autocommit

Para establecer el tiempo de espera de reintento de solicitud en una sesión o para alterar temporalmente la propiedad de cliente requestRetryTimeout, llame al método setRequestRetryTimeout(long) en la interfaz Session.

```
Session sessionA = objectGrid.getSession();
sessionA.setRequestRetryTimeout(30000);
ObjectMap mapA = sessionA.getMap("payroll");
String key = "key:" + j;
mapA.insert(key, "valueA");
```

Ahora, esta sesión utiliza un valor requestRetryTimeout de 30000 milisegundos o 30 segundos, independientemente del valor que se establece en el archivo de propiedades del cliente. Si desea más información sobre la interfaz de la sesión, consulte Utilización de sesiones para acceder a los datos de la cuadrícula.

Resolución de problemas de configuración de XML

De forma ocasional, cuando configure eXtreme Scale, podría encontrar un encontrar inesperado.

Las siguientes secciones especifican varios problemas de configuración de XML que se pueden producir.

Archivos XML de política de despliegue y ObjectGrid no coincidentes

Los archivos XML de política de despliegue y ObjectGrid deben coincidir. Si no tienen nombres de correlaciones y nombres ObjectGrid coincidentes, se producen errores.

Referencias incorrectas a BackingMap y correlaciones

Si la lista de backingMap del archivo XML de ObjectGrid no coincide con la lista de referencias de correlaciones en un archivo XML de política de despliegue, se produce un error en el servidor de catálogo.

Por ejemplo, el siguiente archivo XML de ObjectGrid y archivo XML de política de despliegue se utiliza para iniciar un proceso de contenedor. El archivo de política de despliegue tiene más referencias a correlaciones que se listan en el archivo XML de ObjectGrid.

ObjectGrid.xml - ejemplo incorrecto

```
<?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>
```

deploymentPolicy.xml - ejemplo incorrecto

```
<?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>
```

Mensajes

Se genera un ObjectGridException causado por la excepción de IncompatibleDeploymentPolicyException. Para el ejemplo anterior, la excepción se visualiza como el siguiente mensaje:

com.ibm.websphere.objectgrid.IncompatibleDeploymentPolicyException: la referencia a la correlación de libro de contabilidad en el mapSet mapSet1 de contabilidad objectgridDeployment no hace referencia a una backingMap válida en el XML de ObjectGrid.

Si a la política de despliegue le faltan referencias de correlaciones a backingMaps listadas en el archivo XML de ObjectGrid, también se produce un ObjectGridException causado por la excepción IncompatibleDeploymentPolicyException. Por ejemplo:

com.ibm.websphere.objectgrid.IncompatibleDeploymentPolicyException: el ObjectGrid de contabilidad contiene backingMap de empleado en el XML de ObjectGrid. Esta correlación no se menciona en un mapSet dentro del objectGridDeployment de contabilidad en el XML de política de despliegue.

Problema

La lista de backingMap en las referencias a correlaciones y al archivo XML de ObjectGrid XML de la política de despliegue deben coincidir.

Solución

Determine qué lista es correcta para el entorno y corrija el archivo XML que contiene la lista incorrecta.

Nombres de ObjectGrid incorrectos

El nombre de el ObjectGrid se menciona en el archivo XML de ObjectGrid y el archivo XML de la política de despliegue.

Mensaje

Se genera un ObjectGridException causado por la excepción de IncompatibleDeploymentPolicyException. A continuación se muestra un ejemplo.

Causado por:com.ibm.websphere.objectgrid.IncompatibleDeploymentPolicyException: el objectgridDeployment con el objectGridName accountin no tiene un ObjectGrid correspondiente en el XML de ObjectGrid.

Problema

El archivo XML de ObjectGrid es la lista maestra de nombres de ObjectGrid. Si una política de despliegue tiene un nombre de ObjectGrid que no está incluido en el archivo XML de ObjectGrid, se produce un error.

Solución

Verifique que el nombres de ObjectGrid se haya escrito correctamente. Elimine todos los nombres adicionales, o añada los nombres de ObjectGrid que faltan, a los

archivos XML de ObjectGrid o de política de despliegue. En el mensaje de ejemplo, se ha escrito incorrectamente el objectGridName como ″accountin″, en lugar de ″accounting″.

El valor XML del atributo no es válido

A algunos de los atributos del archivo XML sólo se les puede asignar determinados valores. Estos atributos tienen valores aceptables enumerados por el esquema. La siguiente lista proporciona alguno de los atributos:

- Atributo authorizationMechanism en el elemento objectGrid
- Atributo copyMode en el elemento backingMap
- Atributo lockStrategy en el elemento backingMap
- Atributo ttlEvictorType en el elemento backingMap
- Atributo type en el elemento property
- initialState en el elemento objectGrid
- evictionTriggers en el elemento backingMap

Si se asigna un valor no válido a uno de estos atributos, no se supera la validación XML. En el siguiente archivo XML de ejemplo, se utiliza un valor de INVALID_COPY_MODE incorrecto:

Ejemplo de 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>
```

En el registro aparece este mensaje.

CWOBJ2403E: El archivo XML no es válido. Se ha detectado un problema con < null > en la línea 5. El mensaje de error es 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]'. Debe ser un valor de la enumeración.

Atributos o códigos que faltan

Si a un archivo XML le faltan los atributos o códigos correctos, pueden producirse errores. Por ejemplo, falta el siguiente archivo XML de ObjectGrid en el código de cierre < /objectGrid >:

faltan atributos - XML de ejemplo
```
<?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>
```

En el registro aparece este mensaje.

CWOBJ2403E: El archivo XML no es válido. Se ha detectado un problema con < null > en la línea 7. El mensaje de error es: El código final para el tipo de elemento ″objectGrid″ debe terminar con un delimitador '>'.

Se produce un ObjectGridException acerca del archivo XML no válido con el nombre del archivo XML

Errores de sintaxis

Si un archivo XML se formatea con una sintaxis incorrecta, el mensaje CWOBJ2403E aparece en el registro. Por ejemplo, el siguiente mensaje se visualiza cuando falta una comilla en uno de los atributos XML.

CWOBJ2403E: El archivo XML no es válido. Se ha detectado un problema con < null > en la línea 7. El mensaje de error es: se espera una comilla abierta para el atributo ″maxSyncReplicas″ asociado a un tipo de elemento ″mapSet″.

También se produce un ObjectGridException acerca del archivo XML no válido.

Referencia a una colección de plug-ins no existente

Cuando se utiliza XML para definir plug-ins de BackingMap, el atributo pluginCollectionRef del elemento backingMap debe hacer referencia a un objeto backingMapPluginCollection. El atributo pluginCollectionRef debe coincidir con el ID de uno de los elementos backingMapPluginCollection.

Mensaje

Si el atributo pluginCollectionRef no coincide con ningún atributo de ID de ninguno de los elementos backingMapPluginConfiguration, se mostrará en el archivo de registro el siguiente mensaje o uno similar.

```
[7/14/05 14:02:01:971 CDT] 686c060e XmlErrorHandl E CWOBJ9002E:
Este es un mensaje informativo sólo en inglés: Invalid XML file.
Line: 14; URI: null; Message: Key 'pluginCollectionRef' with
value 'bookPlugins' not found for identity constraint of element 'objectGridConfig'.
```

Problema

Se utiliza el siguiente archivo XML para producir el error. Observe que el nombre del manual BackingMap tiene su atributo pluginCollectionRef establecido en bookPlugins, y la backingMapPluginCollection única tiene un ID de collection1.

```
referencia a un XML de atributo no existente - ejemplo
<?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>
```

Solución

Para corregir el problema, asegúrese de que el valor de cada pluginCollectionRef coincida con el ID de uno de los elementos backingMapPluginCollection. Simplemente cambie el nombre de pluginCollectionRef por collection1 para no recibir este error. De forma alternativa, cambie el ID de la

backingMapPluginCollection existente de modo que coincida con pluginCollectionRef, o añada una backingMapPluginCollection adicional con un ID que coincida con pluginCollectionRef para corregir el error.

Validación de XML sin soporte de implementación

IBM Software Development Kit (SDK) versión 1.4.2 contiene una implementación de alguna función de JAXP (Java API for XML Processing) para utilizar para la validación de XML respecto a un esquema.

Cuando se utiliza un SDK que no contiene esta implementación, los intentos de realizar la validación no serán satisfactorios. Si desea validar el XML utilizando un SDK que no contiene esta implementación, descargue Apache Xerces e incluya sus archivos JAR (Java Archive) en la classpath.

Cuando intente validar XML con un SDK que no tiene la implementación necesaria, el registro contiene el siguiente error:

```
XmlConfigBuil d 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)...
```

El SDK que se utiliza no contiene una implementación de la función JAXP que es necesaria para validar archivos XML con un esquema.

Para evitar este problema, después de descargar Xerces e incluir los archivos JAR en la classpath, podrá validar el archivo de XML satisfactoriamente.

Cargadores

Con un plug-in Loader de eXtreme Scale, una correlación de eXtreme Scale se puede comportar como una memoria caché para los datos que, normalmente, se conservan en un almacén persistente en el mismo sistema o en algún otro. Una base de datos o un sistema de archivos suele utilizarse como el almacén persistente. Una máquina virtual Java (JVM) remota también se puede utilizar como el origen de datos, lo que permite crear memorias caché basadas en hub utilizando eXtreme Scale. Un cargador tiene la lógica para leer y escribir datos en un almacén persistente.

Visión general

Los cargadores son plug-ins de correlaciones de respaldo que se invocan cuando se realizan cambios en la correlación de respaldo o ésta no puede satisfacer una solicitud de datos (una falta de memoria caché). Consulte la información sobre cómo almacenar en memoria caché los escenarios en *Visión general del producto* para obtener una visión general sobre cómo eXtreme Scale puede interactuar con un cargador.

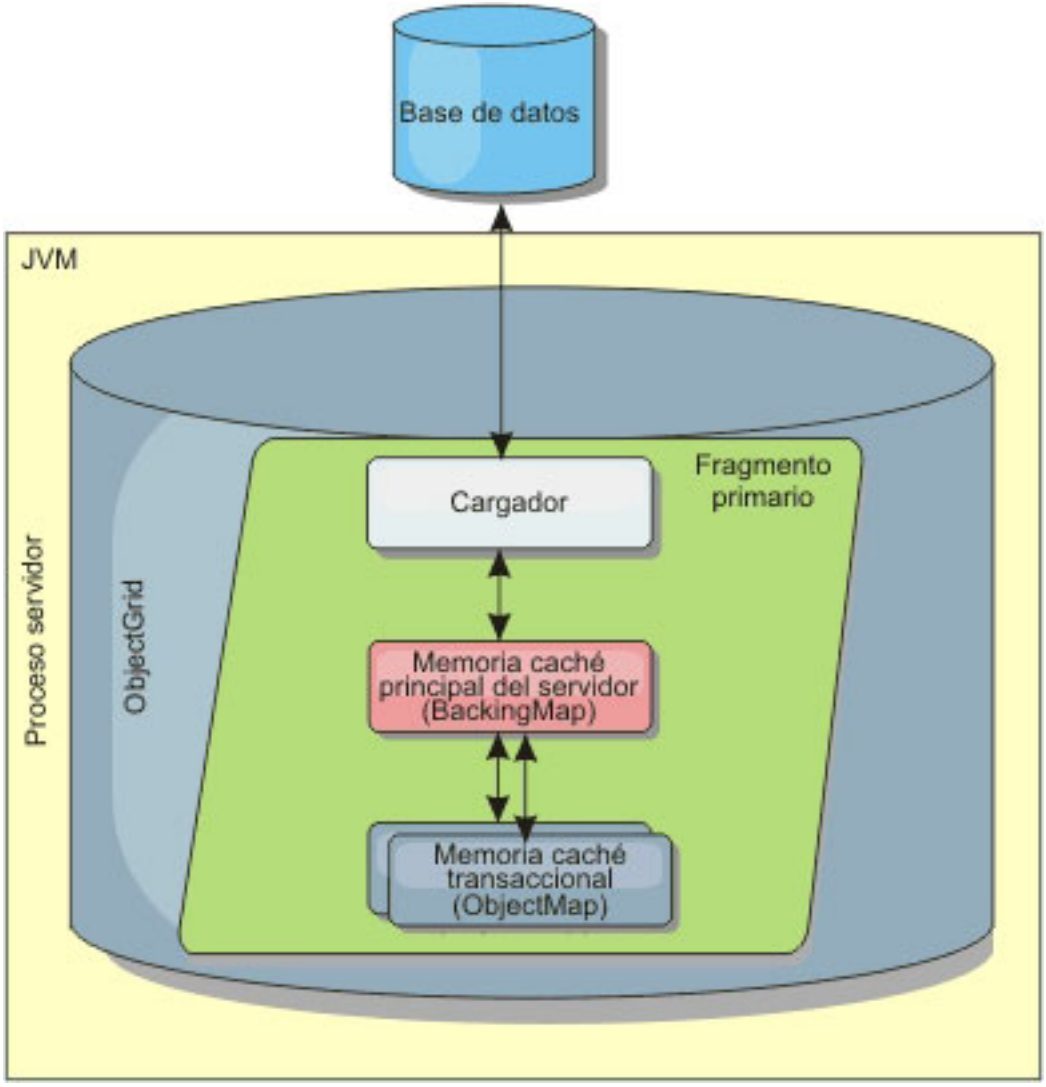


Figura 1. Cargador

Dos cargadores incorporados pueden simplificar en gran medida la integración con los programas de fondo de la base de datos relacional. Los cargadores JPA utilizan las funciones de correlación de objetos relacionales (ORM) de ambas implementaciones, OpenJPA e Hibernate, de la especificación de JPA (Java Persistence API). Consulte la información sobre los cargadores JPA en *Visión general del producto* si desea más información.

Uso de un cargador

Para añadir un cargador a la configuración de BackingMap, puede utilizar la configuración mediante programa o la configuración del archivo XML. Un cargador tiene la siguiente relación con una correlación de respaldo.

- Una correlación de respaldo sólo puede tener un cargador.
- Una correlación de respaldo de cliente (memoria caché cercana) no puede tener un cargador.
- Una definición de cargador se puede aplicar a varias correlaciones de respaldo, pero cada una de éstas tiene su propia instancia de cargador.

Si desea más información, consulte el tema sobre cómo escribir un cargador en *Visión general del producto*.

Conexión de un cargador mediante la configuración del XML

Un cargador proporcionado por la aplicación se puede conectar utilizando un archivo XML. El siguiente ejemplo demuestra cómo conectar el cargador ″MyLoader″ en la correlación de respaldo ″map1″. Debe especificar el className para el cargador, el nombre de la base de datos y los detalles de la conexión, y las propiedades del nivel de aislamiento. Puede utilizar la misma estructura XML si sólo utiliza un cargador previo especificando el nombre de clase del cargador previo, en lugar de un nombre de clase de cargador completo.

Configuración de cargador con 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="grid">
        <backingMap name="map1" pluginCollectionRef="map1" lockStrategy="OPTIMISTIC" />
    </objectGrid>
</objectGrids>
<backingMapPluginCollections>
    <backingMapPluginCollection id="map1">
        <bean id="Loader" className="com.myapplication.MyLoader">
            <property name="dataBaseName"
                      type="java.lang.String"
                      value="testdb"
                      description="database name" />
            <property name="isolationLevel"
                      type="java.lang.String"
                      value="read committed"
                      description="iso level" />
        </bean>
    </backingMapPluginCollection>
</backingMapPluginCollections>
</objectGridConfig>
```

Conexión de un cargador mediante programa

Sólo puede utilizar una configuración programática con cuadrículas locales en memoria. El fragmento de código siguiente muestra cómo conectar un cargador proporcionado por la aplicación en la correlación de respaldo map1 mediante ObjectGrid API:

Configuración programática de un cargador
```
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.BackingMap;
ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid og = ogManager.createObjectGrid( "grid" );
BackingMap bm = og.defineMap( "map1" );
MyLoader loader = new MyLoader();
loader.setDataBaseName("testdb");
loader.setIsolationLevel("read committed");
bm.setLoader( loader );
```

Este fragmento de código presupone que la clase MyLoader es la clase proporcionada por la aplicación que implementa la interfaz com.ibm.websphere.objectgrid.plugins.Loader. Puesto que la asociación de un cargador con una correlación de respaldo no se puede modificar después de que se inicialice el ObjectGrid, el código se debe ejecutar antes de invocar el método initialize de la interfaz ObjectGrid que se está llamando. Se produce una excepción IllegalStateException en una llamada de método setLoader, si se llama después de que se haya producido la inicialización.

El cargador que proporciona la aplicación puede tener propiedades establecidas. En el ejemplo, el cargador MyLoader se utiliza para leer y escribir datos de una

tabla en una base de datos relacional. El cargador debe especificar el nombre de la base de datos y el nivel de aislamiento de SQL. El cargador MyLoader tiene los métodos setDataBaseName y setIsolationLevel que permiten a la aplicación establecer estas dos propiedades de cargador.

Consideraciones sobre los cargadores

En este tema se tratan las consideraciones que deben tenerse al implementar un cargador.

Consideraciones de precarga

Los cargadores son plug-ins de correlaciones de respaldo que se invocan cuando se realizan cambios en la correlación de respaldo o ésta no puede satisfacer una solicitud de datos (una falta de memoria caché). Si desea una visión general sobre cómo eXtreme Scale interactúa con un cargador, consulte la información sobre los escenarios de almacenamiento en memoria caché en línea en *Visión general del producto*.

Cada correlación de respaldo tiene un atributo preloadMode booleano que se establece para indicar si la precarga de una correlación se completa de forma asíncrona. De manera predeterminada, el atributo preloadMode está establecido en false, que indica que la inicialización de la correlación de respaldo no se completa hasta que la precarga de la correlación haya terminado. Por ejemplo, la inicialización de la correlación de respaldo no se completa hasta que se haya devuelto el método preloadMap. Si el método preloadMap lee una gran cantidad de datos de su programa de fondo y los carga en la correlación, puede que tarde en completarse. En ese caso, puede configurar una correlación de respaldo de modo que use una precarga asíncrona de la correlación; para ello, establezca el atributo preloadMode en true. Este valor hace que el código de inicialización de la correlación de respaldo genere una hebra que invoca el método preloadMap, lo que permite que se complete la inicialización de una correlación de respaldo mientras la precarga de la correlación está todavía en curso.

El fragmento de código siguiente ilustra cómo se establece el atributo preloadMode para habilitar la precarga asíncrona:

```
BackingMap bm = og.defineMap( "map1" );
bm.setPreloadMode( true );
```

El atributo preloadMode también puede establecerse mediante un archivo XML, como se muestra en el ejemplo siguiente:

```
<backingMap name="map1" preloadMode="true" pluginCollectionRef="map1"
 lockStrategy="OPTIMISTIC" />
```

TxID y uso de la interfaz TransactionCallback

A los métodos get y batchUpdate de la interfaz Loader se pasa un objeto TxID, que representa la transacción Session que requiere que se ejecute la operación get o batchUpdate. Es posible que la transacción llame más de una vez a los métodos get y batchUpdate. Por lo tanto, los objetos con ámbito de transacción que el cargador necesita se conservan normalmente en una ranura del objeto TxID. Se utiliza un cargador JDBC (Java database connectivity) para ilustrar cómo utiliza un cargador las interfaces TxID y TransactionCallback.

También es posible almacenar en la misma base de datos varias correlaciones ObjectGrid. Cada correlación tiene su propio cargador y cada cargador podría

necesitar conectarse a la misma base de datos. Al conectarse a la misma base de datos, cada cargador desea utilizar la misma conexión JDBC de modo que los cambios de cada tabla se confirmen como parte de la misma transacción de base de datos. Normalmente, la misma persona que escribe la implementación de cargador también escribe la implementación de TransactionCallback. El procedimiento recomendado es, una vez que se ha ampliado la interfaz TransactionCallback, añadir métodos que necesite el cargador para obtener una conexión de base de datos y para almacenar en memoria caché sentencias preparadas. Comprenderá porqué se recomienda este procedimiento en cuanto vea cómo utiliza el cargador las interfaces TransactionCallback y TxID.

En el ejemplo siguiente verá cómo el cargador necesita que la interfaz TransactionCallback se amplíe:

```
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 );
}
```

Al usar estos métodos nuevos, los métodos get y batchUpdate del cargador pueden obtener una conexión de la siguiente forma:

```
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;
}
```

En el ejemplo anterior y en los ejemplos siguientes, ivTcb y ivOcb son variables de instancia del cargador que se inicializaron como se describió en el apartado sobre las consideraciones de precarga. La variable ivTcb es una referencia a la instancia MyTransactionCallback e ivOcb es una referencia a la instancia MyOptimisticCallback. La variable databaseName es una variable de instancia del cargador que se estableció como propiedad de cargador durante la inicialización de la correlación de respaldo. El argumento isolationLevel es una de las constantes JDBC Connection definidas para los diversos niveles de aislamiento que JDBC admite. Si el cargador utiliza una implementación optimista, el método get suele utilizar una conexión JDBC de confirmación automática para captar los datos de la base de datos. En ese caso, el cargador podría tener un método getAutoCommitConnection que se implementase de la siguiente manera:

```
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;
}
```

Recuerde que el método batchUpdate tiene la siguiente sentencia switch:

```
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;
}
```

Cada uno de los métodos buildBatchSQL utiliza la interfaz MyTransactionCallback para obtener una sentencia preparada. A continuación se muestra un fragmento de código que ilustra cómo el método buildBatchSQLUpdate crea una sentencia update de SQL para actualizar una entrada EmployeeRecord y añadirla a la actualización de proceso por lotes:

```
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();
}
```

Una vez que el bucle batchUpdate ha credo todas las sentencias preparadas, llama al método getPreparedStatementCollection. Este método se implementa de la siguiente manera:

```
private Collection getPreparedStatementCollection( TxID tx, Connection conn )
{
    return ( ivTcb.getPreparedStatementCollection( tx, conn, "employee" ) );
}
```

Cuando la aplicación invoca el método commit en Session, el código de Session llama al método commit en el método TransactionCallback después de haber enviado al cargador todos los cambios realizados por la transacción para cada correlación que la transacción modificó. Debido a que todos los cargadores utilizaron el método MyTransactionCallback para obtener las conexiones y las sentencias preparadas que necesitan, el método TransactionCallback sabe qué conexión utilizar para solicitar que el programa de fondo confirme los cambios. Por lo tanto, ampliar la interfaz TransactionCallback con los métodos que necesite cada uno de los cargadores tiene las ventajas siguientes:

- El objeto TransactionCallback encapsula el uso de ranuras de TxID para los datos con ámbito de transacción, y el cargador no requiere información sobre las ranuras de TxID. El cargador sólo necesita saber qué métodos se van a añadir a TransactionCallback mediante la interfaz MyTransactionCallback para las funciones que necesite el cargador.
- El objeto TransactionCallback puede garantizar que la conexión se comparta entre cada cargador que se conecte el mismo programa de fondo, de modo que puede evitarse un protocolo de confirmación de dos fases.
- Con el objeto TransactionCallback, la conexión al programa de fondo se completa a través de una confirmación o retrotracción que se invoca en la conexión cuando se necesita.
- TransactionCallback garantiza que se produzca la limpieza de los recursos de base de datos cuando se completa una transacción.
- TransactionCallback oculta si está obteniendo una conexión gestionada de un entorno gestionado como, por ejemplo, WebSphere Application Server u otro

servidor de aplicaciones compatible con Java 2 Platform, Enterprise Edition (J2EE). Esta ventaja permite que se utilice el mismo código de cargador en entornos gestionados y no gestionados. Sólo debe cambiarse el plug-in TransactionCallback.

- Para obtener más información sobre cómo la implementación TransactionCallback utiliza las ranuras de TxID para los datos con ámbito de transacción, consulte Plug-in TransactionCallback

OptimisticCallback

Como se ha mencionado anteriormente, el cargador puede utilizar un acercamiento optimista para conseguir un control de simultaneidad. En ese caso, el ejemplo del método buildBatchSQLUpdate debe modificarse ligeramente para implementar un acercamiento optimista. Existen diversas formas de utilizar un acercamiento optimista. Puede tener un columna de indicación de hora o una columna de contador de número de secuencia para añadir una versión a cada actualización de la fila. Presuponga que la tabla de empleados tiene una columna de número de secuencia que aumenta cada vez que se actualiza la fila. A continuación, deberá modificar la firma del método buildBatchSQLUpdate de modo que se pase al objeto LogElement en lugar del par clave/valor. También deberá utilizar el objeto OptimisticCallback conectado a la correlación de respaldo para obtener el objeto de versión inicial y para actualizar el objeto de versión. A continuación se muestra un ejemplo de un método buildBatchSQLUpdate modificado que utiliza la variable de instancia ivOcb que se inicializó como se describió en el apartado sobre preloadMap:

ejemplo de código de método de actualización por lotes modificado
```
private void buildBatchSQLUpdate( TxID tx, LogElement le, Connection conn )
 throws SQLException, LoaderException
{
    // Obtener el objeto de versión inicial cuando esta entrada de correlación se leyó
    // o actualizó por última vez en la base de datos.
    Employee emp = (Employee) le.getCurrentValue();
    long initialVersion = ((Long) le.getVersionedValue()).longValue();
    // Obtener el objeto de versión del objeto Employee actualizado para la
    //operación update de SQL.
    Long currentVersion = (Long)ivOcb.getVersionedObjectForValue( emp );
    long nextVersion = currentVersion.longValue();
    // A continuación cree una operación update de SQL que incluya el objeto de
versión en la cláusula where
    // para la comprobación optimista.
    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();
}
```

El ejemplo muestra que se utiliza el objeto LogElement para obtener el valor de versión inicial. Cuando la transacción accede por primera vez a la entrada de correlación, se crea un objeto LogElement con el objeto Employee inicial que se obtiene de la correlación. Este objeto Employee inicial se pasa también al método getVersionedObjectForValue en la interfaz OptimisticCallback y el resultado se guarda en el LogElement. Este proceso se produce antes de que se dé a la aplicación una referencia al objeto Employee inicial, por lo que es posible llamar a algún método que cambie el estado del objeto Employee inicial.

El ejemplo muestra que el cargador utiliza el método getVersiondObjectForValue para obtener el objeto de versión para el objeto Employee actual y actualizado. Antes de llamar al método batchUpdate en la interfaz Loader, eXtreme Scale llama al método updateVersionedObjectForValue en la interfaz OptimisticCallback para

provocar que se genere un objeto de una nueva versión para el objecto Employee actualizado. Una vez que el método batchUpdate vuelve a ObjectGrid, se actualiza el objeto LogElement con el objeto de versión actual y pasa a ser el nuevo objeto de versión inicial. Este paso es necesario porque la aplicación podría haber llamado al método flush en la correlación en lugar del método commit en Session. Es posible que una única transacción llame al cargador varias veces para la misma clave. Por dicho motivo, eXtreme Scale se asegura de que se actualice el objeto LogElement con el objeto de la nueva versión, cada vez que se actualiza la fila en la tabla de empleados.

Ahora que el cargador tiene el objeto de versión inicial y el objeto de versión siguiente, puede ejecutar una sentencia update de SQL que establezca la columna SEQNO en el valor del objeto de versión siguiente y utilice el valor del objeto de versión inicial en la cláusula where. Este procedimiento suele denominarse sentencia de actualización sobrecualificada. El uso de la sentencia de actualización sobrecualificada permite que la base de datos relacional verifique que ninguna otra transacción modificó la fila entre el tiempo en que esta transacción lee los datos de la base de datos y el tiempo en que actualiza la base de datos. Si otra transacción ha modificado la fila, la matriz de números devuelta por la actualización de proceso por lotes indica que ninguna fila se actualizó para esta clave. El cargador debe verificar que la operación update de SQL ha actualizado verdaderamente la fila. Si no se ha actualizado, el cargador muestra una excepción com.ibm.websphere.objectgrid.plugins.OptimisticCollisionException para informar a Session de que se ha producido una anomalía en el método batchUpdate debido a la existencia de más de una transacción simultánea intentando actualizar la misma fila de la tabla de la base de datos. Al recibir esta excepción, Session se retrotrae y la aplicación debe volver a repetir la transacción entera. La explicación es que esta repetición será correcta, de ahí que este procedimiento se llame optimista. El procedimiento optimista funciona mejor si los datos no se modifican con frecuencia o si transacciones simultáneas rara vez intentan actualizar la misma fila.

Es importante que el cargador utilice el parámetro clave del constructor OptimisticCollisionException para identificar qué clave o conjunto de claves provocó la anomalía en el método batchUpdate optimista. El parámetro clave puede ser el propio objeto clave o una matriz de objetos clave si se obtuvo más de una clave en la anomalía de actualización optimista. eXtreme Scale utiliza el método getKey del constructor OptimisticCollisionException para determinar qué entradas de correlación contienen datos obsoletos y han provocado la excepción. Parte del proceso de retrotracción consiste en desalojar de la correlación cada entrada de correlación obsoleta. El desalojo de las entradas obsoletas es necesario para que las transacciones subsiguientes que accedan a la misma clave o claves llamen al método get de la interfaz Loader para renovar las entradas de correlación con los datos actuales de la base de datos.

Otras maneras que tiene un cargador de implementar un procedimiento optimista son:

- No existe columna de indicación de hora ni columna de número de secuencia. En ese caso, el método getVersionObjectForValue de la interfaz OptimisticCallback devuelve simplemente el objeto de valor como versión. Con este procedimiento, el cargador necesita crear una cláusula where que incluya cada uno de los campos del objeto de versión inicial. Este procedimiento no es eficaz, y no todos los tipos de columna se pueden utilizar en la cláusula where de una sentencia update de SQL sobrecualificada. No se suele utilizar este procedimiento.

- No existe columna de indicación de hora ni columna de número de secuencia. No obstante, a diferencia del procedimiento anterior, la cláusula where sólo contiene los campos de valor que ha modificado la transacción. Otro método para detectar qué campos se han modificado consiste en establecer la modalidad de copia en la correlación de respaldo como modalidad CopyMode.COPY_ON_WRITE. Esta modalidad de copia requiere que una interfaz de valor se pase al método setCopyMode en la interfaz BackingMap. BackingMap crea objetos de proxy dinámicos que implementan la interfaz de valor proporcionada. Con esta modalidad de copia, el cargador puede difundir cada valor a un objeto com.ibm.websphere.objectgrid.plugins.ValueProxyInfo. La interfaz ValueProxyInfo tiene un método que permite al cargador obtener la lista de los nombres de atributos modificados por la transacción. Este método permite que el cargador llame a los métodos get en la interfaz de valor de los nombres de atributos para obtener los datos modificados y para crear una sentencia update de SQL que sólo establezca los atributos modificados. A continuación, puede crearse la cláusula where de modo que tenga la columna de claves primarias más cada una de las columnas de atributos modificados. Este procedimiento es mucho más eficaz que el anterior, pero requiere escribir más código en el cargador y puede que la memoria caché de las sentencias preparadas necesite ser de gran tamaños para poder manejar las diferentes permutaciones. Sin embargo, si las transacciones sólo modifican unos pocos atributos, esta limitación no supondría un problema.
- Puede que algunas bases de datos relacionales tengan una API que sirva de ayuda en el mantenimiento automático de los datos de columnas que resulten útiles en la creación optimista de versiones. Consulte la documentación de la base de datos para determinar si existe esta posibilidad.

Configuración de cargadores JPA

Un cargador Java Persistence API (JPA) es una implementación de plug-in de cargador que utiliza JPA para interactuar con la base de datos. Para configurar un cargador, debe configurar los parámetros necesarios y realizar actualizaciones en los archivos de configuración XML.

Antes de empezar

- Debe tener una implementación de JPA como, por ejemplo, Hibernate u OpenJPA.
- La base de datos puede ser cualquier programa de fondo soportado por el proveedor JPA elegido.
- Puede utilizar el plug-in JPALoader al almacenar datos utilizando la API ObjectMap. Utilice el plug-in JPAEntityLoader al almacenar los datos utilizando la API EntityManager.

Por qué y cuándo se efectúa esta tarea

Si desea más información sobre cómo funciona el cargador Java Persistence API (JPA), consulte la información de *Visión general del producto*.

1. Configure los parámetros necesarios que requiere JPA para interactuar con una base de datos.

   Los siguientes parámetros son necesarios. Estos parámetros se configuran en el bean JPALoader o JPAEntityLoader y el bean JPATxCallback.

   - **persistenceUnitName**: especifica el nombre de la unidad de persistencia. Este parámetro es necesario para dos propósitos: para crear una fábrica de JPA EntityManagerFactory, y para localizar los metadatos de la entidad JPA en el archivo `persistence.xml`. Este atributo se establece en el bean JPATxCallback.

- **JPAPropertyFactory**: especifica la fábrica para crear una correlación de propiedad de persistencia para alterar temporalmente las propiedades de persistencia predeterminadas. Este atributo se establece en el bean JPATxCallback. Para establecer este atributo, es necesaria la configuración del estilo Spring.
- **entityClassName**: especifica el nombre de la clase de entidad necesaria para utilizar los métodos JPA como, por ejemplo, EntityManager.persist, EntityManager.find, etc. JPALoader requiere este parámetro, pero el parámetro es opcional para **JPAEntityLoader**. En el caso de JPAEntityLoader, si no se configura un parámetro **entityClassName**, se utiliza la clase de entidad configurada en la correlación de entidad ObjectGrid. Debe utilizar la misma clase para eXtreme Scale EntityManager y el proveedor JPA. Este atributo se establece en el bean JPALoader o JPAEntityLoader.
- **preloadPartition**: especifica la partición en la que se inicia la precarga de la correlación. Si la partición de la carga previa es menor que cero, o mayor que el número total de particiones menos 1, no se inicia la precarga. El valor predeterminado es -1, que significa que la precarga no se inicia de forma predeterminada. Este atributo se establece en el bean JPALoader o JPAEntityLoader.

Aparte de los cuatro parámetros de JPA que se configuran en eXtreme Scale, los metadatos de JPA se utilizan para recuperar la clave de las entidades JPA. Los metadatos JPA se pueden configurar como una anotación, o como un archivo `orm.xml` especificado en el archivo `persistence.xml`. No forman parte de la configuración de eXtreme Scale.

2. Configure los archivos XML para la configuración JPA.

   Para configurar un JPALoader o JPAEntityLoader, consulte la información sobre los plug-ins de cargador en *Guía de programación*.

   Configure una devolución de llamada de transacción JPATxCallback junto con la configuración del cargador. El siguiente ejemplo es un archivo descriptor XML de ObjectGrid (`objectgrid.xml`), que tiene un JPAEntityLoader y JPATxCallback configurados:

   configuración de un cargador incluida la devolución de llamada - ejemplo 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="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>
```

Si desea configurar una JPAPropertyFactory, debe utilizar una configuración del estilo Spring. A continuación aparece un ejemplo de archivo de configuración XML, `JPAEM_spring.xml`, que configura un bean Spring para ser utilizado para las configuraciones de eXtreme Scale.

configuración de un cargador incluida la fábrica de propiedades JPA - ejemplo XML

```
<?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>
```

Sigue el archivo XML de configuración de `Objectgrid.xml`. Observe que el nombre de ObjectGrid es JPAEM, que coincide con el nombre de ObjectGrid en el archivo de configuración Spring `JPAEM_spring.xml`.

configuración del cargador JPAEM - ejemplo de 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="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>
```

Se puede anotar una entidad con las anotaciones JPA y, también, las anotaciones del gestor de entidades eXtreme Scale. Cada anotación tiene un XML equivalente que puede utilizarse. Por lo tanto, eXtreme Scale se añade al espacio de nombres de Spring. También puede configurarlas mediante el uso del soporte de espacio de nombres Spring. Si desea más detalles sobre el soporte del espacio de nombres de Spring, consulte “Integración con la infraestructura Spring” en la página 210.

Configuración de un actualizador de datos basado en la hora de JPA

Puede configurar una actualización de base de datos basada en tiempo utilizando un XML para una configuración de eXtreme Scale local o distribuida. También puede configurar una configuración local a través de programa.

Por qué y cuándo se efectúa esta tarea

Si desea más información sobre cómo trabaja el actualizador de datos basado en la hora de Java Persistence API (JPA), consulte la información de *Guía de programación*.

Cree una configuración de timeBasedDBUpdate.

- **Con un archivo XML:**

  El siguiente ejemplo muestra un archivo `objectgrid.xml` que contiene una configuración de timeBasedDBUpdate:

```
actualizador basado en la hora JPA - ejemplo de 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="changeOG"
            entityMetadataXMLFile="userEMD.xml">
            <backingMap name="user" >
                <timeBasedDBUpdate timestampField="rowChgTs"
                      persistenceUnitName="userderby"
                      entityClass="com.test.UserClass"
                      mode="INVALIDATE_ONLY"
                />
            </backingMap>
        </objectGrid>
    </objectGrids>
    <backingMapPluginCollections>
</objectGridConfig>
```

  En este ejemplo, la correlación ″user″ se configura con una actualización de base de datos basada en tiempo. La modalidad de actualización de base de datos es INVALIDATE_ONLY, y el campo de indicación de hora es rowChgTs.

  Cuando se inicia el ObjectGrid distribuido ″changeOG″ en el servidor de contenedor, se inicia automáticamente la hebra de actualización de base de datos basado en la hora en la partición 0.

- **A través de programa:**

  Si crea un ObjectGrid local, también puede crear un objeto TimeBasedDBUpdateConfig y establecerlo en la instancia de BackingMap:

```
public void setTimeBasedDBUpdateConfig(TimeBasedDBUpdateConfig dbUpdateConfig);
```

  Si desea más información sobre cómo establecer un objeto en la instancia de BackingMap, consulte la información sobre la interfaz BackingMap en la documentación de la API.

  De forma alternativa, puede anotar el campo de indicación de fecha y hora en la clase de entidad utilizando la anotación com.ibm.websphere.objectgrid.jpa.dbupdate.annotation.Timestamp. Al configurar el valor en la clase, no tendrá que configurar el timestampField en la configuración del XML.

Qué hacer a continuación

Inicie el actualizador de datos basado en la hora de JPA. Consulte la información sobre cómo iniciar el actualizador de datos basado en la hora de JPA en *Guía de programación* si desea más información.

Plug-in de memoria caché JPA

WebSphere eXtreme Scale incluye los plug-ins de memoria caché de nivel (L2) para los proveedores OpenJPA e Hibernate Java Persistence API (JPA).

EL uso de eXtreme Scale como un proveedor de memoria caché de nivel 2 aumenta el rendimiento al leer y consultar datos y reduce la carga de la base de datos. WebSphere eXtreme Scale tiene ventajas sobre las implementaciones de memoria caché incorporadas porque la memoria caché se duplica automáticamente

entre todos los procesos. Cuando un cliente almacena en memoria caché un valor, todos los demás clientes son capaces de utilizar el valor almacenado en memoria caché que está localmente en la memoria.

Con los plug-ins de memoria caché de ObjectGrid OpenJPA e Hibernate, podrá crear tres tipos de topología: incorporada, incorporada con particiones y remota.

Topología incorporada

Una topología incorporada crea un servidor eXtreme Scale dentro del espacio de proceso de cada aplicación. OpenJPA e Hibernate leen directamente con la copia en memoria de la memoria caché y escriben en todas las demás copias. Puede mejorar el rendimiento de la escritura utilizando la réplica asíncrona. El rendimiento de esta topología predeterminada es mejor cuando el volumen de datos en caché es lo suficientemente pequeño para caber en un solo proceso.

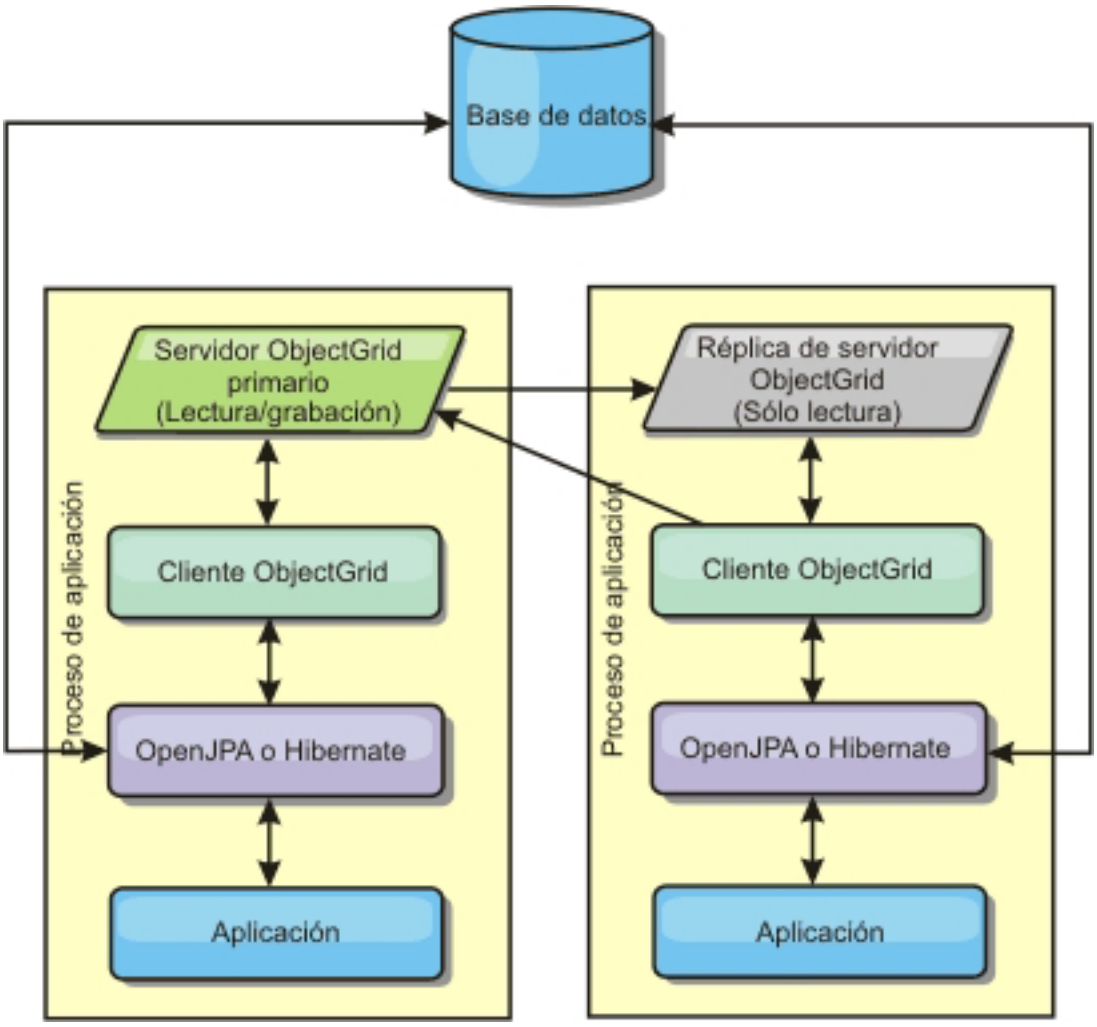


Figura 2. Topología incorporada JPA

Ventajas:

- Todas las lecturas de la memoria caché son muy rápidas, los accesos son locales.
- La configuración es sencilla.

Limitaciones:

- El volumen de los datos se limita al tamaño del proceso.
- Todas las actualizaciones de la memoria caché se envían a un proceso.

Topología incorporada con particiones

Cuando el volumen de los datos de la memoria caché es tan grande que no cabe en un único proceso, la topología incorporada con particiones utiliza las particiones de ObjectGrid para dividir los datos en varios procesos. El rendimiento no es tan alto como el de la topología incorporada porque la mayoría de las lecturas de la memoria caché son remotas. Sin embargo, puede seguir utilizando esta opción cuando la latencia de la base de datos es alta.

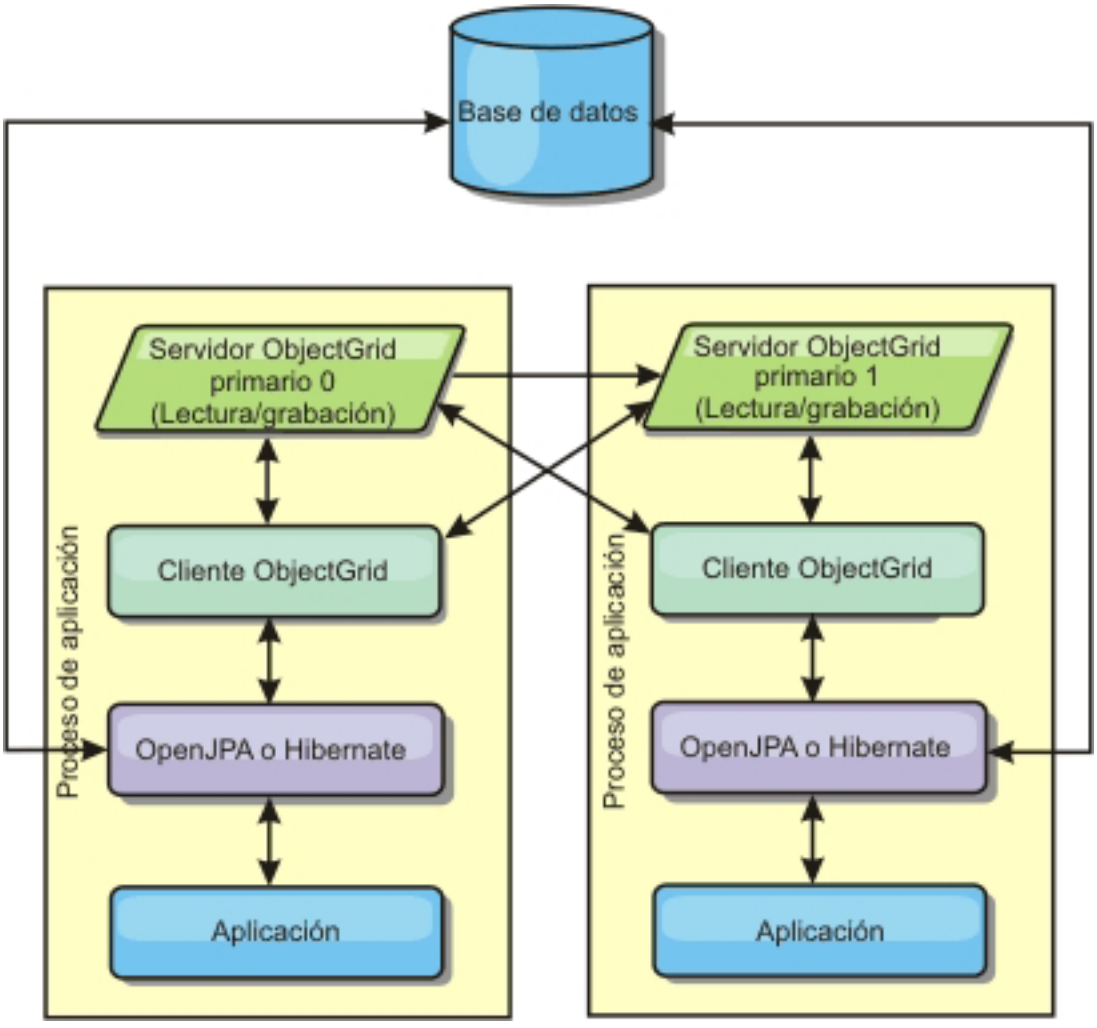


Figura 3. Topología incorporada con particiones JPA

Ventajas:

- Almacena grandes cantidades de datos.
- La configuración es sencilla.
- Las actualizaciones de la memoria caché se reparten en diversos procesos.

Limitación:

- La mayoría de las lecturas y actualizaciones de la memoria caché son remotas.

Por ejemplo, para almacenar en la memoria caché 10 GB de datos con un máximo de 1 GB por JVM, son necesarias diez Máquinas virtuales Java . Por lo tanto, el número de particiones debe establecerse en 10 o más. Lo ideal es establecer el número de particiones en un número primo, donde cada fragmento almacena una cantidad razonable de memoria. Normalmente, el valor numberOfPartitions es igual al número de Máquinas virtuales Java . Con este valor, cada JVM almacena una partición. Si habilita las réplicas, debe aumentar el número de Máquinas virtuales Java en el sistema; de lo contrario, cada JVM también almacena una partición de réplica, que consume tanta memoria como una partición primaria.

Por ejemplo, en un sistema con 4 Máquinas virtuales Java , y el valor de numberOfPartitions de 4, cada JVM aloja una partición primaria. Una operación de lectura tiene un 25 por ciento de posibilidades de captar datos desde una partición disponible localmente, que es mucho más rápido que obtener los datos de una JVM remota. Si una operación de lectura como, por ejemplo, ejecutar una consulta, debe captar una colección de datos que implican 4 particiones de manera uniforme, el 75 por ciento de las llamadas son remotas y el otro 25 por ciento son locales. Si el valor ReplicaMode se establece en `SYNC` o `ASYNC` y el valor ReplicaReadEnabled está establecido en `true`, se crean las cuatro particiones de réplica y se expanden a lo largo de las cuatro Máquinas virtuales Java . Cada JVM aloja una partición primaria y una partición de réplica. La probabilidad de que la operación de lectura se ejecute de forma local aumenta a un 50 por ciento. La operación de lectura que capta una colección de datos que implican cuatro particiones de manera uniforme tiene un 50 por ciento de llamadas remotas y un 50 por ciento de llamadas locales. Las llamadas locales son mucho más rápidas que las memorias remotas. Siempre que se producen llamadas remotas, baja el rendimiento.

Topología remota

Una topología remota almacena todos los datos almacenados en la memoria caché en uno o más procesos separados, reduciendo el uso de la memoria de los procesos de la aplicación. Puede configurar eXtreme Scale que se particione y duplique. Una configuración remota es gestionada independientemente de la aplicación y del proveedor JPA.

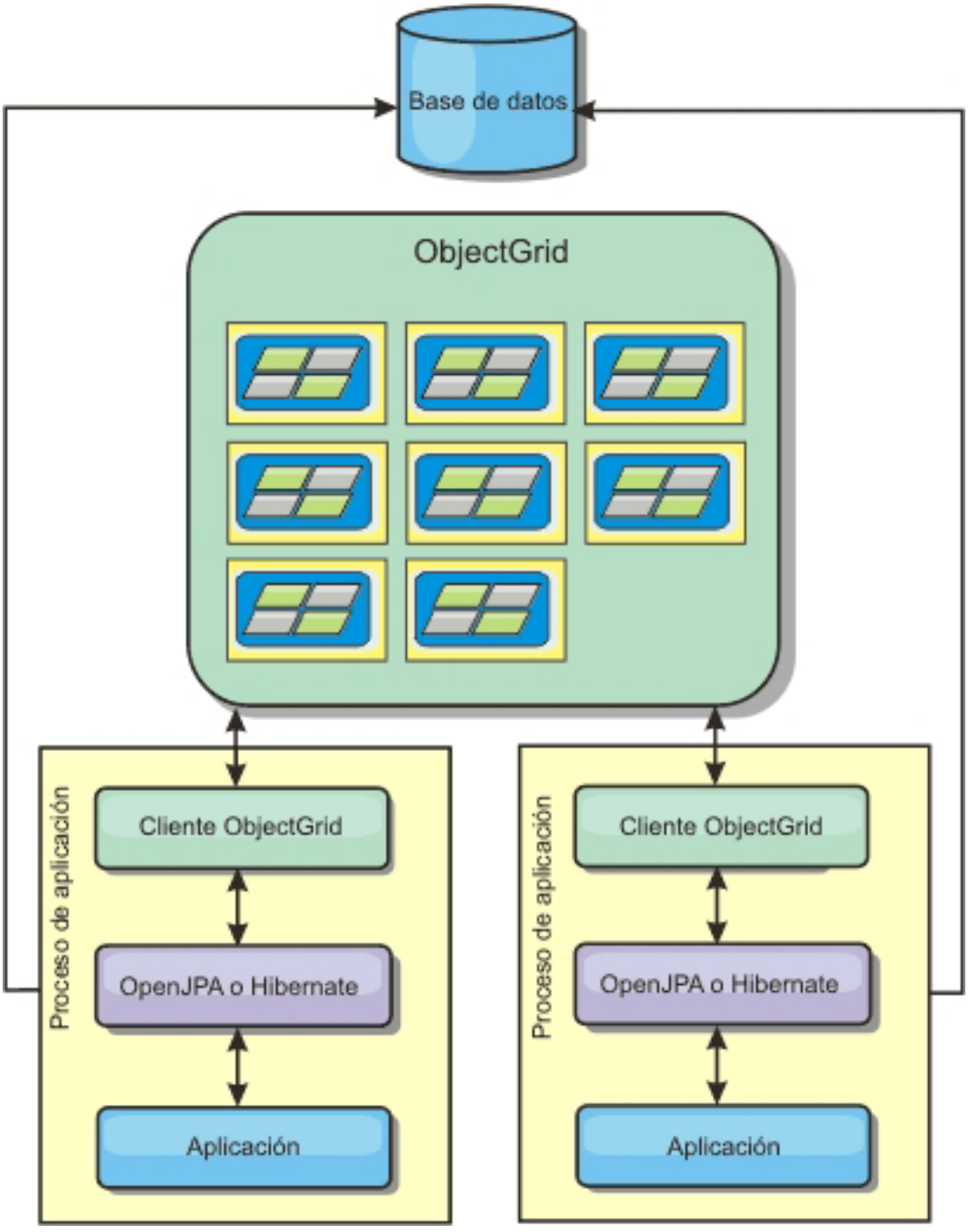


Figura 4. Topología remota JPA

Ventajas:

- Almacena grandes cantidades de datos.
- El proceso de aplicaciones está libre de los datos en memoria caché.
- Las actualizaciones de la memoria caché se reparten en diversos procesos.
- Opciones de configuración muy flexibles.

Limitación:

- Todas las lecturas y actualizaciones de la memoria caché son remotas.

Configuración del plug-in de la memoria caché JPA

WebSphere eXtreme Scale incluye los plug-ins de memoria caché de nivel 2 para los proveedores OpenJPA e Hibernate Java Persistence API (JPA).

Propiedades de configuración de la memoria caché ObjectGrid JPA

Puede configurar el plug-in de memoria caché de JPA con las siguientes propiedades, todas son opcionales.

ObjectGridName

Especifica el nombre ObjectGrid exclusivo. El valor predeterminado es el nombre de la unidad de persistencia definida. Si el nombre de la unidad de persistencia no está disponible desde el proveedor JPA, se utiliza un nombre generado.

ObjectGridType

Especifica el tipo de ObjectGrid.

Valores válidos:

- **EMBEDDED**: es el tipo de configuración predeterminado y recomendado. Sus valores predeterminados incluyen: `NumberOfPartitions=1`, `ReplicaMode=SYNC`, `ReplicaReadEnabled=true` y `MaxNumberOfReplicas=47`. Utilice el parámetro **ReplicaMode** para establecer la modalidad de réplica y el parámetro **MaxNumberOfReplicas** para establecer el número máximo de réplicas. Si un sistema tiene más de 47 Máquinas virtuales Java , establezca el valor **MaxNumberOfReplicas** para que sea igual al número de Máquinas virtuales Java .
- **EMBEDDED_PARTITION**: el tipo para utilizar cuando el sistema necesita almacenar en la memoria caché una gran cantidad de datos en un sistema distribuido. El número predeterminado de particiones es `47` con una modalidad de réplica de `NONE`. En un sistema pequeño que sólo tiene unas pocas Máquinas virtuales Java , establezca el valor **NumberOfPartitions** en un valor igual o menor que el número de Máquinas virtuales Java . Puede especificar los valores **ReplicaMode**, **NumberOfPartitions**, y **ReplicaReadEnabled** para ajustar el sistema.
- **REMOTE**: la memoria caché intenta conectarse a un ObjectGrid remoto distribuido desde el servicio de catálogos.

NumberOfPartitions

Valores válidos: mayor o igual que 1Especifica el número de particiones que se utiliza para la memoria caché. Esta propiedad se aplica cuando el valor ObjectGridType está establecido en `EMBEDDED_PARTITION`. El valor predeterminado es `47`. Para el tipo `EMBEDDED`, el valor de **NumberOfPartitions** siempre es 1.

ReplicaMode

Valores válidos: `SYNC/ASYNC/NONE`Especifica el método que se utiliza para copiar la memoria caché en las réplicas. Esta propiedad se aplica cuando el valor de ObjectGridType está establecido en `EMBEDDED` o `EMBEDDED_PARTITION`. El valor predeterminado es `NONE` para el tipo `EMBEDDED_PARTITION` y `SYNC` para el tipo `EMBEDDED`. Si el valor **ReplicaMode** está establecido en `NONE` para el ObjectType `EMBEDDED`, el tipo `EMBEDDED` sigue utilizando una **ReplicaMode** de `SYNC`.

ReplicaReadEnabled

Valores válidos: TRUE o FALSECuando está habilitado, los clientes leen las réplicas. Esta propiedad se aplica al tipo EMBEDDED_PARTITION. El valor predeterminado es FALSE para el tipo EMBEDDED_PARTITION. El tipo EMBEDDED siempre establece el valor **ReplicaReadEnabled** en TRUE.

MaxUsedMemory

Valores válidos: TRUE o FALSEHabilita el desalojo de las entradas de la memoria caché cuando la memoria se restringe. El valor predeterminado es TRUE y desaloja los datos cuando el umbral de uso del almacenamiento dinámico de la JVM supera el 70 por ciento. Puede modificar el porcentaje predeterminado del umbral de utilización del almacenamiento dinámico de la JVM estableciendo la propiedad memoryThresholdPercentage en el archivo objectGridServer.properties y colocando este archivo en la classpath. Si desea más información sobre los desalojadores, consulte la información sobre los desalojadores en *Visión general del producto*. Si desea más información sobre el archivo de propiedades del servidor, consulte en *Guía de administración*.

MaxNumberOfReplicas

Valores válidos: mayor o igual que 1Especifica el número máximo de réplicas que se debe utilizar para la memoria caché. Este valor sólo se aplica al tipo EMBEDDED. Este número debe ser igual o mayor que el número de Máquinas virtuales Java en un sistema. El valor predeterminado es 47.

Las propiedades NumberOfPartitions, ReplicaMode, ReplicaReadEnabled y MaxNumberOfReplicas son factores de despliegue de ObjectGrid. Las propiedades NumberOfPartitions, ReplicaMode y ReplicaReadEnabled sólo se aplican al tipo EMBEDDED_PARTITION. ReplicaMode y MaxNumberOfReplicas se aplican al tipo EMBEDDED.

Consideraciones sobre los tipos EMBEDDED y EMBEDDED_PARTITION

Los tipos incorporados de ObjectGrid utilizan las propiedades de configuración descritas anteriormente para configurar y desplegar un conjunto de servidores de contenedor de ObjectGrid y un servicio de catálogo, cuando sea necesario. El ciclo de vida de los contenedores está enlazado con la aplicación JPA y se coloca dentro de la classpath de la aplicación. Cuando se inicia una aplicación, el plug-in detecta o iniciar automáticamente un servicio de catálogos, inicia un contenedor y se conecta al servicio de catálogos. El plug-in se comunica con el contenedor ObjectGrid y sus iguales que se ejecutan en otros procesos de servidor de aplicaciones mediante la conexión de cliente.

Cada entidad JPA tiene una correlación de respaldo independiente asignada utilizando el nombre de clase de la entidad. Cada BackingMap tiene los atributos siguientes:

- readOnly="false"
- copyKey="false"
- lockStrategy="NONE"
- copyMode="NO_COPY"

Nota: Cuando se utiliza un valor de ObjectGridType EMBEDDED o EMBEDDED_PARTITION en un entorno Java SE, utilice el método System.exit(0) al final del programa para detener el servidor eXtreme Scale incorporado. De lo contrario, el programa parece que no responde.

Valores predeterminados de ObjectGridType

El valor de ObjectGridType especifica la topología en la que se despliega la memoria caché de ObjectGrid. El tipo predeterminado, y de mejor rendimiento, es EMBEDDED. Las siguientes secciones describen las propiedades predeterminadas para cada uno de los valores ObjectGridType.

Valores predeterminados de la topología de memoria caché ObjectGrid JPA de tipo EMBEDDED

Cuando se utiliza el tipo ObjectGrid `EMBEDDED`, los siguientes valores de propiedad predeterminadas se utilizan si no especifica ningún valor en la configuración:

- **ObjectGridName**: nombre de unidad de persistencia
- **ObjectGridType**: `EMBEDDED`
- **NumberOfPartitions**: `1` (no se puede modificar, si el tipo de ObjectGrid es `EMBEDDED`)
- **ReplicaMode**: `SYNC`
- **ReplicaReadEnabled**: `TRUE` (no se puede modificar cuando el tipo de ObjectGrid es `EMBEDDED`)
- **MaxUsedMemory**: `TRUE`
- **MaxNumberOfReplicas**: `47` (debe ser inferior o igual al número de Máquinas virtuales Java en un sistema distribuido)

Debe especificar un valor ObjectGridName exclusivo para evitar conflictos de denominación. El valor de MaxNumberOfReplicas debe ser igual o mayor que el número total de Máquinas virtuales Java en el sistema.

Topología de memoria caché ObjectGrid de tipo REMOTE

El tipo ObjectGrid REMOTE no requiere ningún valor de propiedad porque ObjectGrid y la policia de despliegue están definidos independientemente de la aplicación JPA. El plug-in de memoria caché JPA se conecta remotamente a un ObjectGrid remoto.

Puesto que toda la interacción con ObjectGrid es remota, esta topología tiene el rendimiento más lento entre todos los tipos ObjectGrid.

Consideraciones y configuración del servicio de catálogo

Cuando se ejecuta en una topología `EMBEDDED` o `EMBEDDED_PARTITION`, el plug-in de memoria caché JPA inicia automáticamente un servicio de catálogos único dentro de uno de los procesos de aplicación, si es necesario. En un entorno de producción, debe crear una cuadrícula de servidor de catálogo. Si desea más información sobre cómo definir un servicio de catálogos, consulte .la información sobre el servicio de catálogos de alta disponibilidad en *Visión general del producto*

Si se ejecuta dentro de un proceso WebSphere Application Server, el plug-in de memoria caché JPA se conecta automáticamente el servicio de catálogos o a la cuadrícula de servicio de catálogos definido para la célula WebSphere Application Server. Si desea más información sobre cómo definir una cuadrícula de servicio de catálogos, consulte la información sobre cómo iniciar el servicio de catálogos en *Guía de administración*.

Si no se ejecutan los servidores dentro de un proceso WebSphere Application Server, los hosts y los puertos de la cuadrícula de servicio de catálogos se especifican utilizando el archivo de propiedades denominado `objectGridServer.properties`. Este archivo se debe almacenar en la classpath de la aplicación y tiene definida la propiedad **catalogServiceEndPoints**. La cuadrícula del servicio de catálogos se inicia independientemente de los procesos de aplicación y se debe iniciar antes de que se inicien los procesos de aplicación.

el formato del archivo `objectGridServer.properties` es el siguiente:

```
PuntosFinalesServicioCatálogos=<nombrehost>:<puerto1>,<nombrehost2>:<puerto2>
```

Configuración del plug-in de la memoria caché Hibernate

Para Hibernate, se puede habilitar la memoria caché de eXtreme Scale estableciendo las propiedades en el archivo de configuración.

Valores

La sintaxis para definir la propiedad en el archivo `persistence.xml` es la siguiente:

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="<property>=<value>,..." />
<property name="objectgrid.hibernate.regionNames" value="<regionName>,.." />
```

La sintaxis para definir la propiedad en el archivo `hibernate.cfg.xml` es la siguiente:

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"><property>=<value>,...</property>
<property name="objectgrid.hibernate.regionNames"><regionName>,...</property>
```

La propiedad provider_class es la propiedad com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider. Para habilitar la memoria caché de consulta, establezca el valor en `true` en la propiedad use_query_cache. Utilice la propiedad objectgrid.configuration para especificar las propiedades de configuración de la memoria caché de eXtreme Scale.

Debe especificar un valor de propiedad ObjectGridName exclusivo para evitar posibles conflictos de denominación. Las otras propiedades de configuración de la memoria caché de eXtreme Scale son opcionales.

La propiedad objectgrid.hibernate.regionNames es opcional y se debe especificar cuando los valores regionNames se definen después de que se haya inicializado la memoria caché de eXtreme Scale. Considere el ejemplo de una clase de entidad que se ha correlacionado con un regionName con la clase de entidad no especificada en el archivo `persistence.xml` o no incluida en el archivo de correlaciones Hibernate. De forma adicional, tiene una anotación Entity. El regionName para esta clase de entidad se resuelve al cargar la clase cuando se inicializa la memoria caché de eXtreme Scale. Otro ejemplo es el método Query.setCacheRegion(String regionName) que se ejecuta después de la inicialización de la memoria caché de eXtreme Scale. En estas situaciones, incluya todos los regionNames dinámicos posibles determinados en la propiedad objectgrid.hibernate.regionNames de forma que la memoria caché de eXtreme Scale puede preparar las BackingMaps para todos los regionNames.

A continuación, se presentan ejemplos de los archivos `persistence.xml` y `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" />
      <property name="objectgrid.hibernate.regionNames" value="queryRegion1, queryRegion2" />
    </properties>
</persistence-unit>
```

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>
```

Precarga de datos en la memoria caché ObjectGrid

Puede utilizar el método preload de la clase ObjectGridHibernateCacheProvider para precargar los datos en la memoria caché de ObjectGrid para una clase de entidad.

Ejemplo 1

Uso de EntityManagerFactory

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("testPU");
ObjectGridHibernateCacheProvider.preload("objectGridName", emf, TargetEntity.class, 100, 100);
```

Ejemplo 2

Uso de SessionFactory

```
org.hibernate.cfg.Configuration cfg = new Configuration();
// utilizar el método addResource, addClass y setProperty de Configuration para preparar
// la configuración necesaria para crear SessionFactor
SessionFactory sessionFactory= cfg.buildSessionFactory();
ObjectGridHibernateCacheProvider.preload("objectGridName", sessionFactory, TargetEntity.class, 100, 100);
```

Nota:

1. En un sistema distribuido, este mecanismo de precarga sólo se puede invocar desde una máquina virtual Java. El mecanismo de precarga no se puede ejecutar de forma simultánea desde varias máquinas virtuales Java.
2. Antes de ejecutar la precarga, debe inicializar la memoria caché de eXtreme Scale creando EntityManager a través de EntityManagerFactory para crear todos los BackingMaps correspondientes; de lo contrario, la precarga forzará la inicialización de la memoria caché con sólo un objeto BackingMap predeterminado para soportar todas las entidades. Esto significa que todas las entidades deberán compartir un objeto BackingMap.

Personalización de la configuración de la memoria caché Hibernate con XML

Para la mayoría de los escenarios, definir las propiedades de memoria caché debería ser suficiente. Para personalizar de forma adicional el ObjectGrid utilizado por la memoria caché, puede proporcionar archivos XML de configuración de ObjectGrid Hibernate en el directorio `META-INF` de forma similar al archivo `persistence.xml`. Durante la inicialización, la memoria caché intentará localizar estos archivos XML y procesarlos si los encuentra.

Existen tres tipos de archivos XML de configuración de ObjectGrid Hibernate: `hibernate-objectGrid.xml` (configuración de ObjectGrid), `hibernate-objectGridDeployment.xml` (política de despliegue) y `hibernate-objectGrid-client-override.xml` (configuración de alteración temporal de ObjectGrid de cliente). En función de la topología configurada de eXtreme Scale, puede proporcionar cualquier de es tos tres archivos XML para personalizar dicha topología.

Para ambos tipos, EMBEDDED y EMBEDDED_PARTITION, puede proporcionar cualquiera de los tres archivos XML para personalizar el ObjectGrid, la política de despliegue y la configuración de alteración temporal de ObjectGrid de cliente.

Para un ObjectGrid REMOTE, la memoria caché no crea ningún ObjectGrid dinámico. La memoria caché sólo obtiene un ObjectGrid de cliente del servicio de catálogo. Sólo puede proporcionar un archivo `hibernate-objectGrid-client-override.xml` para personalizar la configuración de alteración temporal de ObjectGrid de cliente.

1. **Configuración de ObjectGrid**: utilice el archivo `META-INF/hibernate-objectGrid.xml`. Este archivo se utiliza para personalizar la configuración de ObjectGrid para los tipos EMBEDDED y EMBEDDED_PARTITION. Con el tipo REMOTE, se ignora este archivo. De manera predeterminada, cada clase de entidad tiene un regionName asociado (el valor predeterminado es el nombre de la clase de entidad) que se correlaciona con una configuración de BackingMap de nombre regionName dentro de la configuración de ObjectGrid. Por ejemplo, la clase de entidad com.mycompany.Employee tiene un valor predeterminado de regionName asociado como com.mycompany.Employee BackingMap. La configuración predeterminada de BackingMap es readOnly=″false″, copyKey=″false″, lockStrategy=″NONE″ y `copyMode="NO_COPY"`. Puede personalizar algunos BackingMaps con una configuración elegida. La palabra clave reservada ″ALL_ENTITY_MAPS″ se puede utilizar para representar todas las correlaciones, excepto otras correlaciones personalizadas listas en el archivo `hibernate-objectGrid.xml`. Los BackingMaps que no aparecen listados en este archivo `hibernate-objectGrid.xml` utilizan la configuración predeterminada.
2. **Configuración de ObjectGridDeployment**: utilice el archivo `META-INF/hibernate-objectGridDeployment.xml`. Este archivo se utiliza para personalizar la política de despliegue. Al personalizar la política de despliegue, si se proporciona `hibernate-objectGridDeployment.xml`, se descarta la política predeterminada de despliegue. Todos los valores del atributo de la política de despliegues procederán del archivo `hibernate-objectGridDeployment.xml` proporcionado.
3. **Configuración de ObjectGrid de alteración de temporal de cliente**: utilice el archivo `META-INF/hibernate-objectGrid-client-override.xml`. Este archivo se utiliza para personalizar un ObjectGrid de cliente. De manera predeterminada, la memoria caché ObjectGrid aplica una configuración de alteración temporal

de cliente predeterminada que inhabilita la memoria caché cercana. Si la aplicación requiere una memoria caché cercana, puede proporcionar este archivo y especificar numberOfBuckets="xxx". La alteración temporal del cliente predeterminado inhabilita la memoria caché cercana estableciendo numberOfBuckets="0". La memoria caché cercana se puede activar si se restablece el atributo numberOfBuckets en un valor mayor que cero con el archivo hibernate-objectGrid-client-override.xml. La forma en la que trabaja el archivo hibernate-objectGrid-client-override.xml es similar a hibernate-objectGrid.xml: Altera temporalmente o amplía la configuración predeterminada de alteración temporal de ObjectGrid de cliente.

Ejemplos de archivo XML Hibernate ObjectGrid

Los archivos XML Hibernate ObjectGrid deben crearse según la configuración de una unidad de persistencia.

A continuación, aparece un archivo persistence.xml de ejemplo que representa la configuración de una unidad de persistencia:

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>
```

Lo siguiente es el archivo hibernate-objectGrid.xml que coincide con el archivo persistence.xml:

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>
```

Nota: Las correlaciones org.hibernate.cache.UpdateTimestampsCache, org.hibernate.cache.StandardQueryCache y defaultCacheMap son necesarias.

A continuación, aparece el archivo `hibernate-objectGridDeployment.xml` que coincide con `persistence.xml`:

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>
```

Nota: Las correlaciones org.hibernate.cache.UpdateTimestampsCache, org.hibernate.cache.StandardQueryCache y defaultCacheMap son necesarias.

Sistema externo de una memoria caché con el tipo REMOTE de ObjectGrid

Debe configurar un sistema eXtreme Scale externo si desea configurar una memoria caché con un tipo de ObjectGrid REMOTE. Necesita ambos archivos XML de configuración de ObjectGrid y, también, de ObjectGridDeployment que se basan en el archivo `persistence.xml` para configurar un sistema externo. Los archivos XML de configuración de ObjectGrid y ObjectGridDeployment Hibernate que se describen en la sección de ejemplo de archivos XML de ObjectGrid Hibernate también se pueden utilizar para configurar un sistema eXtreme Scale externo.

Un sistema externo ObjectGrid tiene los procesos de servidor de ObjectGrid y servicio de catálogo. Debe iniciar un servicio de catálogo antes de iniciar servidores de contenedor. Consulte los detalles sobre cómo iniciar los servidores eXtreme Scale autónomos y los procesos de contenedor en *Guía de administración* si desea más información.

Resolución de problemas

1. **CacheException: No se ha podido obtener el servidor ObjectGrid**

   Con un ObjectGridType EMBEDDED o EMBEDDED_PARTITION, la memoria caché intenta obtener una instancia de servidor en el tiempo de ejecución de eXtreme Scale. En un entorno de Java Platform, Standard Edition, se iniciará un servidor eXtreme Scale con el servicio de catálogo incorporado. El servicio de catálogo incorporado intenta escuchar en el puerto 2809; si dicho puerto está siendo utilizado por otro proceso, se produce este error. Si se especifican puntos finales de servicio de catálogo, por ejemplo, con el archivo `objectGridServer.properties`, se produce este error si el nombre de sistema principal o el puerto se ha especificado de forma incorrecta.

2. **CacheException: No se ha podido obtener REMOTE ObjectGrid para REMOTE ObjectGrid configurado. objectGridName = [ObjectGridName], PU name = [persistenceUnitName]**

   Se produce este error cuando la memoria caché no puede obtener un ObjectGrid en los puntos finales del servicio de catálogo proporcionado. Normalmente, este error se debe a un nombre de sistema principal o puerto incorrecto.

3. **CacheException: No se pueden tener dos PU [persistenceUnitName_1, persistenceUnitName_2] configuradas con el mismo nombre ObjectGridName [ObjectGridName] de tipo EMBEDDED**

   Esta excepción se produce si tiene una configuración con muchas unidades de persistencia y las memorias caché de estas unidades se configuran con el mismo nombre de ObjectGrid y el ObjectGridType EMBEDDED. Estas configuraciones de unidades de persistencia pueden estar en los mismos archivos `persistence.xml` o en archivos diferentes. Debe verificar que el nombre de ObjectGrid es exclusivo para cada unidad de persistencia cuando el tipo ObjectGridType es EMBEDDED.

4. **CacheException: REMOTE ObjectGrid [ObjectGridName] no incluye los objetos BackingMap obligatorios [mapName_1, mapName_2,...]**

   Con un tipo de ObjectGrid REMOTE, si el ObjectGrid del cliente obtenido no tiene BackingMaps de entidad completos para soportar la memoria caché para la unidad de persistencia, se genera esta excepción. Por ejemplo, si hay cinco clases de entidad listadas en la configuración para la unidad de persistencia,

pero el ObjectGrid obtenido sólo tiene dos BackingMaps. Aunque el ObjectGrid obtenido pueda tener diez BackingMaps, si no se encuentra ninguna de las cinco BackingMaps de entidad necesarios en los diez BackingMaps, se seguirá produciendo esta excepción.

Configuración del plug-in de la memoria caché OpenJPA

WebSphere eXtreme Scale proporciona ambas implementaciones, DataCache y QueryCache, para OpenJPA. La memoria caché de ObjectGrid OpenJPA o la memoria caché de ObjectGrid es un término común para la implementació de DataCache y, también, de QueryCache.

Valores

La memoria caché de eXtreme Scale se habilitada o inhabilitada para OpenJPA definiendo las propiedades de configuración openjpa.DataCache y openjpa.QueryCache en el archivo `persistence.xml`. La sintaxis para definir la propiedad es la siguiente:

```
<property name="openjpa.DataCache"
          value="<object_grid_datacache_class(<property>=<value>,...)"/>
<property name="openjpa.QueryCache"
          value="<object_grid_querycache_class(<property>=<value>,...)"/>
```

Tanto DataCache, como QueryCache pueden tomar las propiedades de memoria caché de eXtreme Scale para configurar la memoria caché utilizada por la unidad de persistencia.

Además de la definición de la memoria caché de eXtreme Scale, la propiedad openjpa.RemoteCommitProvider debe estar establecida en `sjvm`:

```
<property name="openjpa.RemoteCommitProvider" value="sjvm"/>
```

El valor de tiempo de espera especificado con la anotación @DataCache para cada clase de entidad se lleva hasta BackingMap en el que cada entidad se almacena en la memoria caché. Sin embargo, el valor del nombre especificado con la anotación @DataCache es ignorado por la memoria caché de eXtreme Scale. El nombre de clase de entidad totalmente calificado es el nombre de la correlación de la memoria caché.

Los métodos pin y unpin de StoreCache y QueryCache de OpenJPA no están soportados y no realizan ninguna función.

Puede especificar la propiedad ObjectGridName, la propiedad ObjectGridType y otras propiedades relacionadas con la política de despliegue de la lista de propiedades de la clase de memoria caché de ObjectGrid para personalizar la personalizar de la memoria caché. A continuación se muestra un ejemplo:

```
<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"/>
```

Las configuraciones DataCache y QueryCache son independientes entre sí. Puede habilitar cualquiera de estas configuraciones. Sin embargo, si ambas configuraciones están habilitadas, la configuración de QueryCache utiliza la misma

configuración que la configuración de DataCache y se descartan sus propiedades de configuración.

Personalización de la configuración de la memoria caché OpenJPA con XML

Para la mayoría de los escenarios, definir las propiedades de memoria caché de eXtreme Scale debería ser suficiente. Para personalizar de forma adicional el ObjectGrid utilizado por la memoria caché, puede proporcionar los archivos XML de configuración de ObjectGrid OpenJPA en el directorio de META-INF de forma similar al archivo `persistence.xml`. Durante la inicialización de la memoria caché, la memoria caché de ObjectGrid intenta localizar estos archivos XML y procesa los archivos si los encuentra.

Existen tres tipos de archivos XML de configuración de ObjectGrid OpenJPA: los archivos `openjpa-objectGrid.xml` (configuración de ObjectGrid), `openjpa-objectGridDeployment.xml` (política de despliegue) y `openjpa-objectGrid-client-override.xml` (configuración de alteración temporal de ObjectGrid de cliente). En función del tipo de ObjectGrid configurado, puede proporcionar cualquiera de estos tres archivos XML para personalizar el ObjectGrid.

Para ambos tipos, EMBEDDED y EMBEDDED_PARTITION, puede proporcionar cualquiera de estos tres archivos XML para personalizar el ObjectGrid, la política de despliegue y la configuración de alteración temporal de ObjectGrid de cliente.

Para un ObjectGrid REMOTE, la memoria caché de ObjectGrid no crea un ObjectGrid dinámico. En lugar de esto, la memoria caché sólo obtiene un ObjectGrid del cliente en el servicio de catálogo. Sólo puede proporcionar el archivo `openjpa-objectGrid-client-override.xml` para personalizar la configuración de alteración temporal de ObjectGrid de cliente.

1. **Configuración de ObjectGrid**: utilice el archivo `META-INF/openjpa-objectGrid.xml`. Este archivo se utiliza para personalizar la configuración de ObjectGrid para los tipos EMBEDDED y EMBEDDED_PARTITION. Con el tipo REMOTE, se ignora este archivo. De manera predeterminada, cada clase de entidad se correlaciona con su propia configuración de BackingMap cuyo nombre será un nombre de clase de entidad dentro de la configuración de ObjectGrid. Por ejemplo, la clase de entidad com.mycompany.Employee se correlaciona con el objeto BackingMap com.mycompany.Employee. La configuración predeterminada de BackingMap es `readOnly="false"`, `copyKey="false"`, `lockStrategy="NONE"` y `copyMode="NO_COPY"`. Puede personalizar algunos BackingMaps con la configuración que elija. Puede utilizar la palabra clave reservada `ALL_ENTITY_MAPS` para representar todas las correlaciones, excepto otras correlaciones personalizadas listadas en el archivo `openjpa-objectGrid.xml`. Los BackingMaps que no aparecen listados en este archivo `openjpa-objectGrid.xml` utilizan la configuración predeterminado. Si los BackingMaps personalizados no especifican las propiedades o el atributo BackingMaps y estos atributos se especifican en la configuración predeterminada, se aplican los valores de atributo de la configuración predeterminada. Por ejemplo, si se anota una clase de entidad con `timeToLive=30`, la configuración predeterminada de BackingMap para dicha entidad tiene un valor `timeToLive=30`. Si el archivo personalizado `openjpa-objectGrid.xml` también incluye dicho BackingMap, pero no especifica ningún valor timeToLive, el BackingMap personalizado tiene un valor `timeToLive=30` de forma predeterminada. El archivo `openjpa-objectGrid.xml` tiene como objetivo alterar temporalmente o ampliar la configuración predeterminada.

2. **Configuración de ObjectGridDeployment**: utilice el archivo `META-INF/openjpa-objectGridDeployment.xml`. Este archivo se utiliza para personalizar la política de despliegue. Cuando personalice la política de despliegue, si se proporciona el archivo `openjpa-objectGridDeployment.xml`, se descarta la política de despliegue predeterminada. Todos los valores de atributo de política de despliegue proceden del archivo `penjpa-objectGridDeployment.xml` proporcionado.
3. **Configuración de ObjectGrid de alteración temporal de cliente**: utilice el archivo `META-INF/openjpa-objectGrid-client-override.xml`. Este archivo se utiliza para personalizar un ObjectGrid de cliente. De manera predeterminada, la memoria caché ObjectGrid aplica una configuración de alteración temporal de ObjectGrid de cliente predeterminada que inhabilita la memoria caché cercana. Si una aplicación requiere una memoria caché cercana, puede proporcionar este archivo y especificar `numberOfBuckets="xxx"`. La alteración temporal del cliente predeterminado inhabilita la memoria caché cercana estableciendo `numberOfBuckets="0"`. La memoria caché cercana se puede activar al restablecer numberOfBuckets en un valor mayor que 0 con el archivo `openjpa-objectGrid-client-override.xml`. La forma en la que trabaja el archivo `openjpa-objectGrid-client-override.xml` es similar al archivo `openjpa-objectGrid.xml`. Altera temporalmente o amplía la configuración de alteración temporal de ObjectGrid de cliente predeterminada.

Ejemplos de archivos XML OpenJPA ObjectGrid

Los archivos XML OpenJPA ObjectGrid deben crearse según la configuración de la unidad de persistencia.

A continuación, aparece un archivo `persistence.xml` que es un ejemplo que representa la configuración de una unidad de persistencia:

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>
```

A continuación, aparece el archivo `openjpa-objectGrid.xml` que coincide con el archivo `persistence.xml>`

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>
```

Nota:

1. Cada entidad se correlaciona con un BackingMap cuyo nombre es el nombre de clase de entidad totalmente calificado.
2. Si las clases de entidad están en una jerarquía de herencia, las clases hija se correlacionan con el BackingMap padre. La jerarquía de herencia comparte un solo BackingMap.
3. La correlación ObjectGridQueryCache es necesaria para dar soporte a QueryCache.
4. El objeto backingMapPluginCollection de cada correlación de entidad debe tener ObjectTransformer con la clase com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer.
5. El objeto backingMapPluginCollection de la correlación ObjectGridQueryCache debe tener el índice de clave denominado QueryCacheKeyIndex, como se muestra en el ejemplo.
6. El desalojador es opcional para cada correlación.

A continuación, aparece el archivo `openjpa-objectGridDeployment.xml` que coincide con el archivo `persistence.xml`:

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>
```

Nota: La correlación ObjectGridQueryCache es necesaria para dar soporte a QueryCache.

Sistema externo de una memoria caché con el tipo REMOTE de ObjectGrid

Debe configurar un sistema externo si desea configurar una memoria caché con el tipo ObjectGrid REMOTE. Necesita los archivos XML de configuración de ObjectGrid ObjectGridDeployment basados en un archivo `persistence.xml` para configurar un sistema externo. Los archivos XML de configuración de OpenJPA ObjectGrid y ObjectGridDeployment descritos en el apartado de ejemplos de archivo XML de ObjectGrid de OpenJPA también se pueden utilizar para configurar un sistema eXtreme Scale externo.

Un sistema eXtreme Scale externo tiene los procesos del servicio de catálogo y, también, los procesos del servidor de contenedor. Debe iniciar el servidor de catálogo antes de iniciar los servidores de contenedor.

Resolución de problemas

1. **CacheException: No se ha podido obtener el servidor ObjectGrid**

   Con un ObjectGridType del tipo EMBEDDED o EMBEDDED_PARTITION, la memoria caché de eXtreme Scale intenta obtener una instancia de servidor en el tiempo de ejecución. En un entorno Java Platform, Standard Edition, se inicia un servidor eXtreme Scale con el servicio de catálogo incorporado. El servicio de catálogo incorporado intenta escuchar el puerto 2809; si dicho puerto está siendo utilizado por otro proceso, se produce el error. Si se especifican los puntos finales del servicio de catálogo externo, por ejemplo, con el archivo `objectGridServer.properties`, este error se produce si el nombre de sistema principal o el puerto se ha especificado de forma incorrecta.

2. **CacheException: No se ha podido obtener REMOTE ObjectGrid para REMOTE ObjectGrid configurado. objectGridName = [ObjectGridName], PU name = [persistenceUnitName]**

   Este error se produce cuando la memoria caché no puede obtener un ObjectGrid de los puntos finales de servicio de catálogo proporcionado. Normalmente, este error se debe a un nombre de sistema principal o puerto incorrecto.

3. **CacheException: No se pueden tener dos PU [persistenceUnitName_1, persistenceUnitName_2] configuradas con el mismo nombre ObjectGridName [ObjectGridName] de tipo EMBEDDED**

   Esta excepción se genera si tiene muchas configuraciones de unidades de persistencia y las memorias caché de eXtreme Scale de estas unidades se configuran con el mismo nombre de ObjectGrid y un ObjectGridType de EMBEDDED. Estas configuraciones de unidades de persistencia podrían estar en los mismos archivos `persistence.xml` o en archivos diferentes. Debe verificar que el nombre de ObjectGrid es exclusivo para cada unidad de persistencia cuando el tipo ObjectGridType es EMBEDDED.

4. **CacheException: REMOTE ObjectGrid [ObjectGridName] no incluye los objetos BackingMap obligatorios [mapName_1, mapName_2,...]**

   Con un tipo de ObjectGrid REMOTE, si el ObjectGrid del cliente obtenido no tiene BackingMaps de entidad completos para soportar la memoria caché de unidad de persistencia, se produce esta excepción. Por ejemplo, se listan cinco clases de entidad en la configuración de la unidad de persistencia, pero el ObjectGrid obtenido sólo tiene dos BackingMaps. Aunque el ObjectGrid obtenido pueda tener diez BackingMaps, si no se encuentra ninguno de los cinco BackingMaps de entidad necesarios en los diez BackingMaps, se seguirá produciendo esta excepción.

Nota: La memoria caché de OpenJPA de eXtreme Scale ha cambiado el formato de datos para mejorar el rendimiento. Cualquier sistema que incluya aplicaciones OpenJPA configuradas con eXtreme Scale como una memoria caché de nivel 2 se debe detener antes de migrar a WebSphere eXtreme Scale versión 7.0.

Soporte de almacenamiento en memoria caché de grabación diferida

Puede utilizar el almacenamiento en la memoria caché de grabación diferida para reducir la sobrecarga que se produce al actualizar una base de datos de programa de fondo. El almacenamiento en memoria caché de grabación diferida pone en cola las actualizaciones del plug-in Loader.

Introducción

El almacenamiento en memoria caché de grabación diferida pone en cola de forma asíncrona actualizaciones del plug-in de cargador (Loader). Puede mejorar el rendimiento mediante la desconexión de actualizaciones, inserciones y eliminaciones de una correlación, la sobrecarga de la actualización de la base de datos de programa de fondo. La actualización asíncrona se realiza después de un retardo basado en la hora (por ejemplo, cinco minutos) o un retardo basado en entradas (1000 entradas).

Cuando configura el valor de grabación diferida en una correlación de respaldo, se crea una hebra de grabación diferida y envuelve el cargador configurado. Cuando una transacción de eXtreme Scale inserta, actualiza o elimina una entrada de una correlación eXtreme Scale, se crea un objeto LogElement para cada uno de estos registros. Estos elementos se envían al cargador de grabación diferida y se ponen en cola en un objeto ObjectMap especial llamado correlación de cola. Cada correlación de respaldo con el valor de grabación diferida habilitado tiene sus propias correlaciones de cola. Una hebra de grabación diferida elimina periódicamente los datos en cola de las correlaciones de cola y los empuja al cargador de programa de fondo real.

El cargador de grabación diferida sólo enviará los tipos de inserción, actualización y eliminación de los objetos LogElement al cargador real. Todos los demás tipos de objetos LogElement, por ejemplo el tipo EVICT, se pasan por alto.

Ventajas

La habilitación del soporte de grabación diferida tiene las ventajas siguientes:

- Aislamiento de errores de programa de fondo: el almacenamiento en memoria caché de grabación diferida proporciona una capa de aislamiento de los errores de programa de fondo. Cuando la base de datos de programa de fondo falla, las actualizaciones se ponen en cola en la correlación de cola. Las aplicaciones pueden continuar con las transacciones a eXtreme Scale. Cuando se recupera el programa de fondo, los datos de la correlación de cola se empujan al programa de fondo.
- Carga reducida de programa de fondo: el cargador de grabación diferida fusiona las actualizaciones de acuerdo con una clave, de modo que sólo existe una actualización fusionada por clave en la correlación de cola. Este procedimiento de fusión disminuye el número de actualizaciones al programa de fondo.
- Rendimiento mejorado de transacciones: los tiempos individuales de las transacciones de eXtreme Scale disminuyen porque la transacción no necesita esperar a que los datos se sincronicen con el programa de fondo.

XML de descriptor ObjectGrid

Cuando se configura un eXtreme Scale utilizando un archivo XML de descriptor de eXtreme Scale, el cargador de grabación diferida se habilita estableciendo el atributo writeBehind en el código de backingMap. A continuación se muestra un ejemplo:

```
<objectGrid name="library" >
     <backingMap name="book" writeBehind="T300;C900" pluginCollectionRef="bookPlugins"/>
```

En el ejemplo anterior, el soporte de grabación diferida de la correlación de respaldo ″book″ está habilitado con el parámetro ″T300;C900″.

El atributo de grabación diferida especifica el tiempo de actualización máximo y/o un recuento máximo de actualizaciones de claves. El formato del parámetro de grabación diferida es:

```
atributo de grabación diferida  ::= <predeterminado> | <hora actualización> | <recuento claves actualización> |
<hora actualización> ";" <recuento claves actualización>
hora actualización ::= "T" <entero positivo>
recuento claves actualización ::= "C" <entero positivo>
valores predeterminados  ::= "" {table}
```

Las actualizaciones en el cargador se producen cuando se produce uno de los siguientes sucesos:

1. Ha transcurrido el tiempo máximo de actualización en segundos desde la última actualización.
2. El número de claves actualizadas en la correlación de colas ha alcanzado el recuento de claves de actualización.

Estos parámetros sólo son sugerencias. El recuento de actualizaciones y la hora de actualización reales estarán en un rango cercano de parámetros. Sin embargo, no puede garantizar que el recuento real de actualizaciones o la hora de actualización sean los mismos que se han definido en los parámetros. Además, la primera actualización diferida podría darse hasta con dos veces más de tiempo que la hora de actualización. Esto se debe a que eXtreme Scale elige aleatoriamente la hora de inicio de la actualización para que así las particiones no accedan simultáneamente a la base de datos.

En el ejemplo anterior T300;C900, el cargador escribe los datos en el programa de fondo cuando han transcurrido 300 segundos, después de la última actualización o cuando hay 900 claves pendientes para ser actualizadas.

La hora de actualización predeterminada es de 300 segundos y el recuento de claves de actualización predeterminado.

La tabla siguiente lista algunos ejemplos de atributo de escritura diferida.

Nota: Si configura el cargador de grabación diferida como una serie vacía: `writeBehind=""`, el cargador de grabación diferida se habilita utilizando los valores predeterminados. Por lo tanto, no especifique el atributo writeBehind si no desea que el soporte de grabación anticipada esté habilitado.

Tabla 6. Algunas opciones de escritura diferida

Valor de atributo	Hora
T100	La hora de actualización es 100 segundos y el recuento de claves de actualización predeterminado es 1000 (el valor predeterminado)
C2000	La hora de actualización es 300 segundos (el valor predeterminado) y el recuento de claves de actualización es 2000.
T300;C900	La hora de actualización es 300 segundos y el recuento de claves de actualización es 900.
""	La hora de actualización es 300 segundos (el valor predeterminado) y el recuento de claves de actualización es 1000 (el valor predeterminado).

Habilitación mediante programación del soporte de grabación diferida

Al crear una correlación de respaldo mediante un programa para un eXtreme Scale local y en memoria, puede utilizar el siguiente método en la interfaz BackingMap para habilitar o inhabilitar el soporte de grabación diferida.

```
public void setWriteBehind(String writeBehindParam);
```

Si desea más detalles sobre cómo utilizar el método setWriteBehind, consulte “Interfaz BackingMap” en la página 67.

Consideraciones sobre el diseño de aplicaciones

Habilitar el soporte de grabación diferida es sencillo, pero diseñar una aplicación que funcione con el soporte de grabación diferida requiere un cuidado especial. Sin el soporte de grabación diferida, la transacción eXtreme Scale encierra la transacción del programa de fondo. La transacción de eXtreme Scale se inicia antes de que se inicie la transacción de programa de fondo, pero termina después de que termine la transacción de programa de fondo.

Con el soporte de grabación diferida habilitado, la transacción de eXtreme Scale finaliza antes de que se inicie la transacción de programa de fondo. La transacción de eXtreme Scale y la transacción de programa de fondo se desacoplan.

Restricciones de la integridad referencial

Cada correlación de respaldo que se configura con soporte de grabación diferida tiene su propia hebra de grabación diferida que empuja los datos al programa de fondo. Por lo tanto, los datos que se actualizan en correlaciones diferentes de una transacción de eXtreme Scale se actualizan en el programa de fondo en diferentes transacciones de programa de fondo. Por ejemplo, la transacción T1 actualiza la clave key1 en la correlación Map1 y la clave key2 en la correlación Map2. La actualización de key1 en la correlación Map1 se actualiza en el programa de fondo en una transacción de programa de fondo, y la clave key2 actualizada en la correlación Map2 se actualiza en el programa de fondo en otra transacción de programa de fondo mediante distintas hebras de grabación diferida. Si los datos almacenados en Map1 y Map2 tienen relaciones, como restricciones de clave foránea en el programa de fondo, puede que se produzca un error en las actualizaciones.

Al diseñar las restricciones de la integridad referencial en la base de datos de programa de fondo, asegúrese de que se permiten las actualizaciones que no funcionan.

Actualizaciones anómalas

Puesto que la transacción de eXtreme Scale termina antes de que se inicie la transacción de programa de fondo, es posible que se produzca un éxito falso de la transacción. Por ejemplo, si intenta insertar una entrada en una transacción de eXtreme Scale que no existe en la correlación de respaldo, pero existe en el programa de fondo, lo que genera una clave duplicada, la transacción de eXtreme Scale se realiza correctamente. Sin embargo, la transacción en la que la hebra de grabación diferida inserta ese objeto en el programa de fondo sufre una anomalía con una excepción de clave duplicada.

Consulte “Manejo de actualizaciones de grabación diferida erróneas” en la página 127 para ver cómo manejar dichas anomalías.

Comportamiento de bloqueo de correlaciones de cola

Otra diferencia principal en el comportamiento de las transacciones es el comportamiento de bloqueo. eXtreme Scale da soporte a tres estrategias de bloqueo distintas: PESSIMISTIC, OPTIMISITIC y NONE. Las correlaciones de colas de grabación diferida utiliza la estrategia de bloqueo pesimista sin importar qué

estrategia de bloqueo está configurada para su correlación de soporte. Hay dos tipos distintos de operaciones que adquieren un bloqueo en la correlación de cola:

- Cuando una transacción de eXtreme Scale se confirma, o se produce una operación de desecho (desecho de correlación o desecho de sesión), la transacción lee la clave en la correlación de colas y pone un bloqueo S en la clave.
- Cuando se confirma una transacción de eXtreme Scale, la transacción intenta actualizar el bloqueo S por el bloqueo X en la clave.

Debido a este comportamiento de correlación de colas adicional, puede ver algunas diferencias en el comportamiento del bloqueo.

- Si la correlación de usuarios está configurada como estrategia de bloqueo pesimista (PESSIMISTIC), no hay mucha diferencia de comportamiento en el bloqueo. Cada vez que se llama a una operación de desecho o confirmación, se coloca un bloqueo S en la misma clave de la misma correlación de colas. Durante la confirmación, no sólo se adquiere un bloqueo X para la clave en la correlación de usuarios, sino que además de adquiere para la clave en la correlación de colas.
- Si la correlación de usuarios está configurada como estrategia de bloqueo optimista (OPTIMISTIC) o ninguna (NONE), la transacción de usuario seguirá el patrón de estrategia de bloqueo pesimista (PESSIMISTIC). Cada vez que se llama a una operación de desecho o confirmación, se adquiere un bloqueo S en la misma clave de la misma correlación de colas. Durante la confirmación se adquiere un bloqueo X para la clave en la correlación de colas utilizando la misma transacción.

Reintentos de transacción de cargador

WebSphere eXtreme Scale no soporta las transacciones de 2 fases o XA. La hebra de grabación diferida elimina registros de la correlación de colas y actualiza los registros en el programa de fondo. Si se produce una anomalía en el servidor durante la transacción, puede que se pierdan algunas actualizaciones del programa de fondo.

El cargador de grabación diferida reintentará automáticamente grabar las transacciones fallidas y enviará un LogSequence dudosa al programa de fondo para evitar la pérdida de datos. Esta acción requiere que el cargador sea idempotent, lo que significa que cuando se llama al método Loader.batchUpdate(TxId, LogSequence) dos veces con el mismo valor, da el mismo resultado que si se hubiera aplicado una vez. Las implementaciones del cargador deben implementar la interfaz RetryableLoader para habilitar esta característica. Consulte en la documentación de la API si desea más detalles.

Anomalías del cargador

El plug-in de cargador puede fallar cuando no puede comunicarse con el programa de fondo de la base de datos. Esto puede suceder si el servidor de bases de datos o la conexión de red está inactivo. El cargador de grabación diferida pondrá en cola las actualizaciones e intentará empujar los cambios de los datos al cargador de forma periódica. El cargador debe notificar al tiempo de ejecución de WebSphere eXtreme Scale que existe un problema de conectividad de la base de datos lanzando una excepción LoaderNotAvailableException.

Por lo tanto, la implementación del cargador debe distinguir entre una anomalía de datos o un anomalía física del cargador. La anomalía de datos se debe lanzar o

volver a lanzar como una excepción LoaderException o OptimisticCollisionException, pero una anomalía del cargador físico se debe lanzar o volver a lanzar como una excepción LoaderNotAvailableException. WebSphere eXtreme Scale maneja estas dos excepciones de forma distinta:

- Si el cargador de grabación diferida obtiene una excepción LoaderException, el cargador de grabación diferida considerará la anomalía como un error de los datos, como por ejemplo un error de clave duplicada. El cargador de grabación diferida anulará el proceso por lotes de la actualización, e intentará actualizar un registro cada vez para aislar la anomalía de los datos. Si se vuelve a obtener una excepción LoaderException durante la actualización de un registro, se crea un registro de actualización con errores y se anota en la correlación de actualizaciones con errores.
- Si el cargador de grabación diferida obtiene una excepción LoaderNotAvailableException, el cargador de grabación diferida la considerará como un error porque no puede conectase a la base de datos, por ejemplo, el programa de fondo de la base de datos está inactivo, una conexión de base de datos no está disponible, o la red no está activa. El cargador de grabación diferida esperará 15 segundos y después volverá a intentar realizar la actualización por lotes en la base de datos.

El error habitual es emitir una excepción LoaderException cuando debería emitirse una excepción LoaderNotAvailableException. Todos los registros puestos en cola en el cargador de grabación diferida pasan a ser registros de actualizaciones con anomalías, que anula el propósito del aislamiento de anomalías de programa de fondo. Probablemente, se producirá este error si escribe un cargador genérico para hablar con las bases de datos.

El JPALoader proporcionado por eXtreme Scale es un ejemplo. El JPALoader utiliza la API de JPA para interactuar con los programas de fondo de base de datos. Cuando falla la red, JPALoader obtiene una javax.persistence.PersitenceException, pero desconoce el motivo de la anomalía, a menos que se comprueben el estado de SQL y el código de error SQL de la SQLException encadenada. El hecho de que el JPALoader se ha diseñado para trabajar con todos los tipos de base de datos complica más el problema, porque los estados y los códigos de error de SQL son diferentes para el problema de caída de red. Para resolver esto, WebSphere eXtreme Scale proporciona una API ExceptionMapper para permitir a los usuarios conectar una implementación para correlacionar una Exception con una excepción más consumible. Por ejemplo, los usuarios pueden correlacionar una javax.persistence.PersitenceException genérica con una LoaderNotAvailableException, si el código de error o el estado de SQL indica que se ha caído la red.

Consideraciones sobre el rendimiento

El soporte de almacenamiento en memoria caché de grabación diferida aumenta el tiempo de respuesta al eliminar la actualización del cargador de la transacción. Aumenta además el rendimiento de la base de datos ya que las actualizaciones de las bases de datos se combinan. Es importante comprender la sobrecarga que introduce la hebra de grabación diferida, que extrae los datos de la correlación de cola y los envía al cargador.

El número máximo de actualizaciones o el tiempo máximo de actualización debe ajustarse en función del entorno y de los patrones de uso esperados. Si el valor del número máximo de actualizaciones o el tiempo máximo de actualización es demasiado pequeño, la sobrecarga de la hebra de grabación diferida puede

sobrepasar las ventajas. Si se especifica un valor elevado para estos dos parámetros, podría aumentarse el uso de memoria al poner en cola los datos y aumentarse el tiempo obsoleto de los registros de la base de datos.

Para obtener un rendimiento óptimo, ajuste los parámetros de grabación diferida de acuerdo con los factores siguientes:

- Índice de transacciones de lectura y grabación.
- Misma frecuencia de actualización de registros.
- Latencia de actualización de la base de datos.

Configuración del soporte de grabación diferida

Puede habilitar el soporte de grabación diferida utilizando el archivo XML de descriptor de ObjectGrid, o a través de programa utilizando la interfaz BackingMap.

Utilice el archivo XML de descriptor ObjectGrid para habilitar el soporte de grabación diferida, o a través de programa mediante la interfaz BackingMap.

Archivo XML de descriptor ObjectGrid

Cuando se configura un ObjectGrid utilizando un archivo XML de descriptor de ObjectGrid, el cargador de grabación diferida se habilita estableciendo el atributo writeBehind en el código backingMap. A continuación se muestra un ejemplo:

```
<objectGrid name="library" >
     <backingMap name="book" writeBehind="T300;C900" pluginCollectionRef="bookPlugins"/>
```

En el ejemplo anterior, el soporte de grabación diferida de la correlación de respaldo book se habilita con el parámetro T300;C900. El atributo de grabación diferida especifica el tiempo de actualización máximo y/o un recuento máximo de actualizaciones de claves. El formato del parámetro de grabación diferida es:

```
::= <valor predeterminado> | <hora actualización> | <recuento de claves de actualización> |
<hora actualización> ";" <recuento claves actualización>::= "T"
<entero positivo>::= "C" <entero positivo>::= ""
```

- atributo de grabación diferida
- hora de actualización
- recuento claves actualización
- valores predeterminados

Las actualizaciones en el cargador se producen cuando se produce uno de los siguientes sucesos:

1. Ha transcurrido el tiempo máximo de actualización en segundos desde la última actualización.
2. El número de claves actualizadas en la correlación de colas ha alcanzado el recuento de claves de actualización.

Estos parámetros sólo son sugerencias. El recuento de actualizaciones y la hora de actualización reales estarán en un rango cercano de parámetros. Sin embargo, no se garantiza que el recuento de actualizaciones real o la hora de actualización sean los mismos que se han definido en los parámetros. Además, la primera actualización diferida podría darse hasta con dos veces más de tiempo que la hora de actualización. Esto se debe a que ObjectGrid elige aleatoriamente la hora de inicio de la actualización para que todas las particiones no accedan a la base de datos simultáneamente.

En el ejemplo anterior T300;C900, el cargador escribe los datos en el programa de fondo cuando han transcurrido 300 después de la última actualización o cuando hay 900 claves pendientes para actualizar. La hora de actualización predeterminada es de 300 segundos y el recuento de claves de actualización predeterminado.

Manejo de actualizaciones de grabación diferida erróneas

Puesto que la transacción de WebSphere eXtreme Scale termina antes de que se inicie la transacción de programa de fondo, es posible que se produzca un éxito falso de la transacción.

Por ejemplo, si intenta insertar una entrada en una transacción de eXtreme Scale que no existe en la correlación de respaldo, pero sí existe en el programa de fondo, lo que produce una clave duplicada, la transacción de eXtreme Scale es correcta. Sin embargo, la transacción en la que la hebra de grabación diferida inserta el objeto en el programa de fondo falla con una excepción de clave duplicada.

Manejo de las actualizaciones de grabación diferida con errores: cliente

Una actualización de este tipo, o cualquier otra actualización de programa de fondo con errores, es una actualización de grabación diferida con errores. Estas actualizaciones de grabación diferida con errores se almacenan en una correlación de actualizaciones de grabación diferida con errores. Esta correlación sirve como cola de sucesos para actualizaciones con errores. La clave de la actualización es un objeto Integer exclusivo, y el valor es una instancia del elemento FailedUpdateElement. La correlación anómala de actualización de escritura diferida se ha configurado con un desalojador, que desaloja los registros 1 hora después de que se hayan insertado. Así pues, los registros de actualización anómala se perderán si no se recuperan en un plazo da una hora.

La API ObjectMap puede utilizarse para recuperar las entradas de correlación de actualizaciones de grabación diferida con errores. El nombre de esta correlación de actualizaciones es: IBM_WB_FAILED_UPDATES_<nombre de correlación>. Consulte la documentación de la API WriteBehindLoaderConstants para conocer los nombre de los prefijos de cada correlación de sistema de grabación diferida. Lo que aparece a continuación es un ejemplo.

proceso anómalo - código de ejemplo

```
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);
    // Realizar alguna acción con la clave, el valor o la excepción.

}
session.commit();
```

Una llamada de getNextKey funciona con una partición específica para cada transacción de eXtreme Scale. En un entorno distribuido, para obtener las claves de todas las particiones, debe iniciar varias transacciones, tal como se muestra en el siguiente ejemplo:

obtención de claves de todas las particiones - código de ejemplo

```
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);
        // Realizar alguna acción con la clave, el valor o la excepción.

    }
    Session.commit();
}
```

Nota: La correlación de actualizaciones con errores proporciona una forma de supervisar la salud de la aplicación. Si un sistema genera numerosos registros en la correlación de actualizaciones con errores, es señal de que debe volver a evaluarse o revisar la arquitectura o aplicación de modo que utilice el soporte de grabación diferida. A partir de 6.1.0.5, puede usar el script xsadmin para ver el tamaño de la entrada de la correlación de actualizaciones con errores.

Manejo de las actualizaciones de grabación diferida con errores: escucha de fragmentos

Es importante detectar y anotar cuándo falla una transacción de grabación diferida. Cada aplicación que utilice la grabación diferida debe implementar un vigilante que maneje las actualizaciones de grabación diferida con errores. Esto evitará que el sistema se quede sin memoria ya que los registros en la correlación de actualizaciones con errores no se desalojan porque se espera que la aplicación los maneje.

El siguiente código muestra cómo conectar dicho vigilante o ″dumper″, que se debe añadir al XML de descriptor de ObjectGrid como en el fragmento de código.

```
<objectGrid name="Grid">

        <bean id="ObjectGridEventListener" className="utils.WriteBehindDumper"/>
```

Puede ver el bean ObjectGridEventListener que se ha añadido, que es el vigilante de grabación diferida mencionado anteriormente. El vigilante interactúa en las correlaciones de todos los fragmentos primarios de una JVM en busca de los que tengan habilitada la grabación diferida. Si encuentra uno, intenta anotar hasta 100 actualizaciones con errores. Sigue vigilando un fragmento primario hasta que éste se mueve a otra JVM . Todas las aplicaciones que usan grabación diferida *deben* usar un vigilante similar a éste. De lo contrario, las máquinas virtuales Java se quedan sin memoria porque esta correlación de errores nunca se desaloja.

Consulte Código de ejemplo de la clase dumper de grabación diferida si desea más información.

Código de ejemplo de la clase de volcador de grabación diferida

Este código fuente de ejemplo muestra cómo escribir un observador (volcador) para manejar actualizaciones de grabación diferida anómalas.

```
//
//Este programa de ejemplo se proporciona TAL CUAL y se puede utilizar, ejecutar, copiar y
//modificar sin que el cliente tenga que pagar derechos (a) para su propia formación,
//(b) para desarrollar aplicaciones diseñadas para ejecutarse con un producto IBM
//WebSphere, ya sea para uso interno propio del cliente o para su redistribución
//por parte del cliente, como parte de una aplicación de este tipo, en los productos propios del cliente. "
//
//5724-J34 (C) COPYRIGHT International Business Machines Corp. 2009
//Reservados todos los derechos * Material bajo licencia - Propiedad de 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;

/**
 * La grabación diferida espera que las transacciones para Loader sean satisfactorias. Si una
  * transacción para una clave falla, insertará una entrada en una correlación denominada
 * PREFIJO + nombreCorrelación. La aplicación debe comprobar si en esta correlación hay
 * entradas para volcar anomalías de transacciones de grabación anticipada. La aplicación es
 * responsable de analizar y luego eliminar estas entradas. Estas entradas pueden ser de gran
 * tamaño porque incluyen la clave, las imágenes del valor de antes y después, y la propia
 * excepción. Las excepciones pueden ocupar fácilmente 20k.
 *
 * La clase se registra con la cuadrícula y se crea una instancia por fragmento primario en una
 * JVM. Crea una única hebra y dicha hebra comprobará cada correlación de errores de grabación
 * diferida para el fragmento, imprimirá el problema y eliminará la entrada.
*
 * Esto significa que habrá una hebra por fragmento. Si el fragmento se traslada a otra JVM, el
 * método deactivate detiene la hebra.
 * @author bnewport
 *
 */
public class WriteBehindDumper implements ObjectGridEventListener, ObjectGridEventGroup.ShardEvents, Callable<Boolean>
{
 static Logger logger = Logger.getLogger(WriteBehindDumper.class.getName());

 ObjectGrid grid;

 /**
  * Agrupación de hebras para manejar verificadores de tablas. Si la aplicación tiene una
  * agrupación propia, cámbiela para reutilizar la agrupación existente
  */
 static ScheduledExecutorService pool = new ScheduledThreadPoolExecutor(2); // dos hebras para volcar registros

 // el futuro para este fragmento
 ScheduledFuture<Boolean> future;

 // true si este fragmento está activo
 volatile boolean isShardActive;

 /**
  * Tiempo normal entre las comprobaciones de correlaciones para ver si hay errores de grabación
  * diferida
  */
 final long BLOCKTIME_SECS = 20L;

 /**
  * Una sesión asignada para este fragmento. No tiene sentido en asignarla una y otra vez
  */
 Session session;
 /**
  * Cuando un fragmento primario se activa, planificar las comprobaciones de forma periódica
  * para comprobar las correlaciones de errores de grabación diferida e imprimir problemas
  */
 public void shardActivated(ObjectGrid grid)
 {
  try
  {
   this.grid = grid;
   session = grid.getSession();

   isShardActive = true;
   future = pool.schedule(this, BLOCKTIME_SECS, TimeUnit.SECONDS); // comprobar cada BLOCKTIME_SECS segundos inicialmente
  }
  catch(ObjectGridException e)
  {
```

```
  throw new ObjectGridRuntimeException("Exception activating write dumper", e);
 }
}

/**
 * Marcar fragmento como inactivo y luego cancelar el verificador
 */
public void shardDeactivate(ObjectGrid arg0)
{
 isShardActive = false;
 // si se cancela, la cancelación devuelve true
 if(future.cancel(false) == false)
 {
  // si no, bloquear hasta que se complete el verificador
  while(future.isDone() == false) // esperar a que la tarea finalice de una forma u otra
  {
   try
   {
    Thread.sleep(1000L); // comprobar cada segundo
   }
   catch(InterruptedException e)
   {
   }
  }
 }
}

/**
 * Prueba simple para ver si la correlación está habilitada para la grabación diferida, y si lo
 * está, devolver el nombre de la correlación de errores para la misma.
 * @param mapName La correlación que se va a probar
 * @return El nombre de la correlación de errores de grabación diferida si existe, si no nulo
 */
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;
}

/**
 * Se ejecuta para cada fragmento. Comprueba si cada correlación tiene habilitada la grabación
  * diferida y a continuación imprime cualquier error de transacción de grabación
 * y, a continuación, elimina el registro.
 */
public Boolean call()
{
 logger.fine("Called for " + grid.toString());
 try
 {
  // mientras el fragmento primario está presente en esta JVM
  // aquí sólo se devuelven las correlaciones definidas por el usuario, en esta lista no hay
  // ningún correlaciones del sistema como correlaciones de grabación diferida
  Iterator<String> iter = grid.getListOfMapNames().iterator();
  boolean foundErrors = false;
  // iterar en todas las correlaciones actuales
  while(iter.hasNext() && isShardActive)
  {
   String origName = iter.next();

   // si es una correlación de errores de grabación diferida
   String name = getWriteBehindNameIfPossible(grid, origName);
   if(name != null)
   {
    // intentar eliminar bloques de N errores cada vez
    ObjectMap errorMap = null;
    try
    {
     errorMap = session.getMap(name);
    }
    catch(UndefinedMapException e)
    {
     // durante el inicio, las correlaciones de errores pueden todavía no existir, paciencia...
     continue;
    }
```

```
    // intentar volcar N registros a la vez
    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);
      //
      // La aplicación debe anotar el problema aquí
      logger.info("WriteBehindDumper ( " + origName + ") for key (" + elem.getKey() + ") Exception: " +
       elem.getThrowable().toString());
      //
      //
      errorMap.remove(seqKey);
     }
     else
      break;
    }
    session.commit();
   }
  } // ejecutar correlación siguiente
  // realice un bucle más rápido si hay errores
  if(isShardActive)
  {
   // volver a planificar después de un segundo si había registro anómalos
   // de lo contrario, espere 20 segundos.
   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();

  //no dejar una transacción en la sesión.
  if(session.isTransactionActive())
  {
   try { session.rollback(); } catch(Exception e2) {}
  }
 }
 return true;
}

public void destroy() {
 // Apéndice de método generado automáticamente TODO

}

public void initialize(Session arg0) {
 // Apéndice de método generado automáticamente TODO

}

public void transactionBegin(String arg0, boolean arg1) {
 // Apéndice de método generado automáticamente TODO

}

public void transactionEnd(String arg0, boolean arg1, boolean arg2,
  Collection arg3) {
 // Apéndice de método generado automáticamente TODO

}
}
```

Configuración de desalojadores

Los desalojadores se pueden configurar utilizando el archivo XML de descriptor de ObjectGrid o a través de programas.

Por qué y cuándo se efectúa esta tarea

Si desea información sobre la configuración con XML, consulte "Archivo XML de descriptor ObjectGrid" en la página 157.

Habilitación del desalojador TTL

WebSphere eXtreme Scale proporciona un mecanismo predeterminado para desalojar entradas de memoria caché y un plug-in para crear desalojadores personalizados. Un desalojador controla la pertenencia de entradas en cada instancia de BackingMap. El desalojador predeterminado utiliza una política de desalojo de tiempo de vida (TTL) para cada instancia de BackingMap. Si proporciona un mecanismo de desalojador conectable, generalmente utiliza una política de desalojo basada en el número de entradas en lugar del tiempo.

Habilitación mediante programación del desalojador TTL

Los desalojadores TTL están asociados a las instancias BackingMap. El siguiente fragmento de código demuestra cómo puede utilizarse la interfaz BackingMap para establecer los atributos necesarios de modo que cuando se crean todas las entradas, éstas tienen una hora de caducidad establecida en diez minutos después de su creación.

habilitación del desalojador de tiempo de vida a través de programa
```
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 );
```

El argumento del método setTimeToLive es 600 porque indica que el tiempo de vida está en segundos. El código anterior se debe ejecutar antes de que se invoque el método initialize en la instancia de ObjectGrid. Estos atributos BackingMap no se pueden modificar después de que se inicialice la instancia de ObjectGrid. Después de ejecutar el código, cualquier entrada que se inserte en la BakingMap myMap tiene una hora de caducidad. Una vez que se ha alcanzado la hora de caducidad, el desalojador TTL elimina la entrada.

Si una aplicación requiere que la hora de caducidad se establezca en la hora del último acceso más diez minutos, se debe modificar una línea del código anterior. El argumento que se pasa al método setTtlEvictorType se cambia de TTLType.CREATION_TIME a TTLType.LAST_ACCESS_TIME. Con este valor, la hora de caducidad se calcula como la hora del último acceso más diez minutos. Cuando se crea una entrada por primera vez, la hora del último acceso es la hora de creación.

Cuando se utiliza el argumento TTLType.LAST_ACCESS_TIME, las interfaces ObjectMap y JavaMap pueden utilizarse para alterar temporalmente el valor de tiempo de vida de BackingMap. Este mecanismo permite que una aplicación utilice un valor de tiempo de vida distinto para cada entrada que se cree. Suponga que el fragmento de código anterior se ha utilizado para establecer el atributo ttlType en LAST_ACCESS_TIME y el valor de tiempo de vida se ha establecido en diez minutos en la instancia de BackingMap. Una aplicación entonces puede alterar temporalmente el valor del tiempo de vida de todas las entradas ejecutando el siguiente código antes de crear o modificar una entrada:

```
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" );
```

En el fragmento de código anterior, la entrada con la clave key1 tiene una fecha de caducidad del tiempo de inserción más 30 minutos como resultado de la invocación del método setTimeToLive(1800) en ObjectMap. La variable oldTimeToLive1 se establece en 600 ya que el valor de tiempo de vida de BackingMap se utiliza como valor predeterminado si no se ha llamado previamente al método setTimeToLive en ObjectMap.

La entrada con la clave key2 tiene una hora de caducidad del tiempo de inserción más 20 minutos como resultado de la invocación del método setTimeToLive(1200 en ObjectMap. La variable oldTimeToLive2 se establece en 1800 ya que el valor de tiempo de vida de la invocación anterior del método ObjectMap.setTimeToLive ha establecido el tiempo de vida en 1800.

El ejemplo anterior muestra la inserción de dos entradas de correlación en la correlación myMap para los valores de clave key1 y key2. Posteriormente, puede que la aplicación de una nueva hebra desee actualizar estas entradas de correlación con nuevos valores de correlación. No obstante, la aplicación desea mantener los valores de tiempo de vida que se utilizan en el momento de la inserción para cada entrada de correlación. El siguiente ejemplo muestra cómo mantener los valores de tiempo de vida utilizando una constante definida en la interfaz ObjectMap:

```
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();
```

Dado que el valor especial de ObjectMap.USE_DEFAULT se utiliza en la llamada al método setTimeToLive, la clave key1 mantiene su valor de tiempo de vida de 1800 segundos y la clave key2 retiene su valor de tiempo de vida de 1200 segundos porque estos valores se utilizaron cuando la transacción anterior insertó estas entradas de correlaciones.

El ejemplo anterior también muestra una nueva entrada de correlación para la inserción de la clave key3nsert. En este caso, el valor especial USE_DEFAULT indica que se debe utilizar el valor predeterminado del tiempo de vida para esta correlación. El valor predeterminado lo define el atributo BackingMap del tiempo de vida. Consulte Consulte los atributos de la interfaz BackingMap si desea más información sobre cómo se define el atributo tiempo de vida en la instancia de BackingMap.

Consulte la documentación de la API para el método setTimeToLive en las interfaces ObjectMap y JavaMap. La documentación le avisa que se genera una excepción IllegalStateException si el método BackingMap.getTtlEvictorType() devuelve cualquier cosa distinta al valor TTLType.LAST_ACCESS_TIME. ObjectMap y JavaMap sólo pueden utilizarse para alterar temporalmente el valor de tiempo de vida cuando se utiliza el tipo de desalojador TTL LAST_ACCESS_TIME TTL. Este método no puede utilizarse para alterar temporalmente el valor de tiempo de vida cuando utilice el tipo de desalojador TTL CREATION_TIME o NONE.

Enfoque de configuración XML para habilitar el desalojador TTL

En lugar de utilizar la interfaz BackingMap para establecer mediante programa los atributos de BackingMap que pueden ser utilizados por el desalojador TTL, puede utilizar un archivo XML para configurar cada instancia de BackingMap. El siguiente código demuestra cómo establecer estos atributos para tres correlaciones de BackingMap distintas:

```
habilitación del desalojador de tiempo de vida mediante 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="grid1">
        <backingMap name="map1" ttlEvictorType="NONE" />
        <backingMap name="map2" ttlEvictorType="LAST_ACCESS_TIME" timeToLive="1800" />
        <backingMap name="map3" ttlEvictorType="CREATION_TIME" timeToLive="1200" />
    </objectgrid>
</objectGrids>
```

El ejemplo anterior muestra que la BackingMap map1 utiliza un tipo de desalojo NONE TTL. La BackingMap map2 utiliza el tipo de desalojador LAST_ACCESS_TIME TTL y tiene un valor de tiempo de vida de 1800 segundos, o 30 minutos. La BackingMap map3 está definida para que utilice un tipo de desalojador TTL CREATION_TIME y tiene un valor de tiempo de vida de 1200 segundos o 20 minutos.

Desalojadores conectables opcionales

El desalojador TTL predeterminado utiliza una política de desalojo basada en la hora, y el número de entradas en BackingMap no tiene ningún efecto en la hora de caducidad de una entrada. Puede utilizar un desalojo conectable opcional para desalojar entradas basándose en el número de entradas que existían en lugar de basarse en la hora.

Los siguientes desalojadores conectables opcionales proporcionan algunos algoritmos utilizados habitualmente para decidir qué entradas se van a desalojar cuando una BackingMap aumenta de tamaño por encima de ciertos límites.

- LRUEvictor es un desalojador que utiliza un algoritmo de menos usada recientemente (LRU) para decidir qué entradas se van a desalojar cuando la BackingMap exceda un número máximo de entradas.
- LFUEvictor es un desalojador que utiliza un algoritmo de usada menos frecuentemente (LFU) para decidir qué entradas se van a desalojar cuando la BackingMap exceda un número máximo de entradas.

BackingMap informa a un desalojador de la creación, modificación o eliminación de entradas en una transacción. BackingMap hace un seguimiento de estas entradas y elige cuándo desalojar una o más entradas de la instancia de BackingMap.

Una instancia de BackingMap no tiene información de configuración para un tamaño máximo. Por el contrario, las propiedades de desalojador se establecen para controlar el comportamiento del desalojador. Tanto LRUEvictor como LFUEvictor tienen una propiedad de tamaño máximo que se utiliza para que el desalojador empiece a desalojar entradas después de que se supere el tamaño máximo. Como el desalojador TTL, es posible que los desalojadores LRU y LFU no desalojen inmediatamente una entrada cuando se alcance el número máximo de entradas para minimizar el impacto en el rendimiento.

Si los algoritmos de desalojo LRU o LFU no son adecuados para una determinada aplicación, puede escribir sus propios desalojadores para crear la estrategia de desalojo.

Utilización de desalojadores conectables opcionales

Para añadir desalojadores conectables opcionales a la configuración de BackingMap puede utilizar la configuración programática o la configuración XML.

Conexión de un desalojador conectable mediante programación

Dado que los desalojadores están asociados a BackingMaps, utilice la interfaz BackingMap para especificar el desalojador conectable. El siguiente fragmento de código es un ejemplo para especificar un desalojador LRUEvictor para la BackingMap map1 y un desalojador LFUEvictor para la instancia del BackingMap map2:

```
conexión de un desalojador a través de programa
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);
```

El fragmento anterior muestra un desalojador LRUEvictor utilizado para la BackingMap map1 con un número máximo de entradas aproximado de 53.000 (53 * 1000). El desalojador LFUEvictor se utiliza para la BackingMap map2 con un número máximo aproximado de entradas de 422.000 (211 * 2000). Los desalojadores LRU y LFU tienen una propiedad de tiempo de inactividad que indica durante cuánto tiempo está inactivo el desalojador antes de entrar en actividad y comprobar si es necesario desalojar alguna entrada. El tiempo de inactividad se especifica en segundos. Un valor de 15 segundos constituye un buen término medio entre el impacto en el rendimiento y cómo evitar que BackingMap crezca demasiado. El objetivo es utilizar el mayor tiempo de inactividad posible sin provocar que BackingMap aumente excesivamente de tamaño.

El método setNumberOfLRUQueues establece la propiedad de LRUEvictor que indica cuántas colas LRU utiliza el desalojador para gestionar la información LRU. Una colección de colas se utiliza de modo que cada entrada no mantenga la información LRU en la misma cola. Este enfoque puede mejorar el rendimiento al minimizar el número de entradas de correlación que necesiten sincronizarse en el mismo objeto de cola. Un buen modo de minimizar el impacto que el desalojador LRU puede tener en el rendimiento consiste en aumentar el número de colas. Un buen punto de partida es utilizar el diez por ciento del número máximo de entradas como número de colas. Generalmente es mejor utilizar número primo que la utilización de uno que no lo sea. El método setMaxSize indica cuántas entradas se permiten en cada cola. Cuando una cola alcanza su número máximo de entradas, la entrada o las entradas menos utilizadas recientemente en esa cola se desalojarán la próxima vez que el desalojador compruebe si hay alguna entrada que es necesario desalojar.

El método setNumberOfHeaps establece la propiedad LFUEvictor para establecer cuántos objetos de almacenamiento dinámico binario utiliza LFUEvictor para gestionar información de LFU. En este caso también se utiliza una colección para mejorar el rendimiento. La utilización del diez por ciento del número máximo de entradas es un buen punto de partida, y un número primo es generalmente mejor que uno que no lo sea. El método setMaxSize indica cuántas entradas se permiten en cada almacenamiento dinámico. Cuando un almacenamiento dinámico alcanza su número máximo de entradas, la entrada o las entradas menos utilizadas frecuentemente en ese almacenamiento dinámico se desalojarán la próxima vez que el desalojador compruebe si hay alguna entrada que es necesario desalojar.

Enfoque de configuración XML para conectar un desalojador conectable

En lugar de utilizar varias API para conectar un desalojador y establecer sus propiedades, se puede utilizar un archivo XML para configurar todas las BackingMap como se ilustra en el siguiente ejemplo:

conexión de un desalojador mediante 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="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>
```

Desalojo basado en memoria

Todos los desalojadores incorporados dan soporte al desalojo basado en memoria que puede habilitarse en la interfaz BackingMap estableciendo el atributo evictionTriggers de BackingMap en ″MEMORY_USAGE_THRESHOLD″. Para obtener más información sobre cómo establecer el atributo evictionTriggers en BackingMap, véala consulta de configuración de la interfaz BackingMap y eXtreme Scale.

El desalojo basado en memoria se basa en el umbral de uso del almacenamiento dinámico. Cuando el desalojo basado en memoria está habilitado en BackingMap y BackingMap tiene cualquier desalojo incorporado, el umbral de uso se establece en un porcentaje predeterminado de la memoria total si el umbral no se ha establecido previamente.

Para cambiar el porcentaje del umbral de uso predeterminado, establezca la propiedad memoryThresholdPercentage en el archivo de propiedades del contenedor y servidor para el proceso de servidor eXtreme Scale. Para establecer el

umbral de uso de destino en un proceso de cliente de eXtreme Scale, puede utilizar MemoryPoolMXBean. Consulte también el archivo containerServer.props y el inicio de procesos de servidor de Xtreme Scale.

Durante el tiempo de ejecución si el uso de memoria excede el umbral del uso del destino, los desalojadores basados en memoria empezarán a desalojar entradas e intentarán mantener el uso de la memoria por debajo del umbral de uso del destino. Sin embargo, no hay ninguna garantía de que la velocidad de desalojo sea lo suficientemente rápida como para evitar un error de falta de memoria posible si el tiempo de ejecución del sistema sigue consumiendo rápidamente la memoria.

Conexión de un desalojador conectable

Dado que los desalojadores están asociados a BackingMaps, utilice la interfaz BackingMap para especificar el desalojador conectable.

Conexión de un desalojador conectable mediante programación

El siguiente fragmento de código es un ejemplo para la especificación de un desalojador LRUEvictor para la BackingMap map1 y un desalojador LFUEvictor para la BackingMap map2:

```
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);
```

El fragmento anterior muestra un desalojador LRUEvictor utilizado para la BackingMap map1 con un número máximo de entradas aproximado de 53.000 (53 * 1000). El desalojador LFUEvictor se utiliza para la BackingMap map2 con un número máximo aproximado de entradas de 422.000 (211 * 2000). Los desalojadores LRU y LFU tienen una propiedad de tiempo de inactividad que indica durante cuánto tiempo está inactivo el desalojador antes de entrar en actividad y comprobar si es necesario desalojar alguna entrada. El tiempo de inactividad se especifica en segundos. Un valor de 15 segundos constituye un buen término medio entre el impacto en el rendimiento y cómo evitar que BackingMap crezca demasiado. El objetivo es utilizar el mayor tiempo de inactividad posible sin provocar que BackingMap aumente excesivamente de tamaño.

El método setNumberOfLRUQueues establece la propiedad de LRUEvictor que indica cuántas colas LRU utiliza el desalojador para gestionar la información LRU. Una colección de colas se utiliza de modo que cada entrada no mantenga la información LRU en la misma cola. Este enfoque puede mejorar el rendimiento al minimizar el número de entradas de correlación que necesiten sincronizarse en el

mismo objeto de cola. Un buen modo de minimizar el impacto que el desalojador LRU puede tener en el rendimiento consiste en aumentar el número de colas. Un buen punto de partida es utilizar el diez por ciento del número máximo de entradas como número de colas. Generalmente es mejor utilizar un número primo que uno que no lo sea. El método setMaxSize indica cuántas entradas se permiten en cada cola. Cuando una cola alcanza su número máximo de entradas, la entrada o las entradas menos utilizadas recientemente en esa cola se desalojarán la próxima vez que el desalojador compruebe si hay alguna entrada que es necesario desalojar.

El método setNumberOfHeaps establece la propiedad LFUEvictor para establecer cuántos objetos de almacenamiento dinámico binario utiliza LFUEvictor para gestionar información de LFU. En este caso también se utiliza una colección para mejorar el rendimiento. La utilización del diez por ciento del número máximo de entradas es un buen punto de partida, y un número primo es generalmente mejor que uno que no lo sea. El método setMaxSize indica cuántas entradas se permiten en cada almacenamiento dinámico. Cuando un almacenamiento dinámico alcanza su número máximo de entradas, la entrada o las entradas menos utilizadas frecuentemente en ese almacenamiento dinámico se desalojarán la próxima vez que el desalojador compruebe si hay alguna entrada que es necesario desalojar.

Configuración del XML para conectar un desalojador conectable

En lugar de utilizar varias API para conectar un desalojador y establecer sus propiedades, se puede utilizar un archivo XML para configurar todas las BackingMap como se ilustra en el siguiente ejemplo:

```
<?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>
```

Configuración de HashIndex

El plug-in HashIndex soporta ambas interfaces, MapIndex y MapRangeIndex. La definición y el uso correctos de índices puede mejorar significativamente el rendimiento de las consultas.

Se puede utilizar los siguientes atributos para configurar HashIndex utilizando el archivo XML de descriptor de despliegue de ObjectGrid o un enfoque programático:

- Nombre: el nombre del índice. El nombre debe ser exclusivo para cada correlación. El nombre se utiliza para recuperar el objeto de índice de la instancia ObjectMap para BackingMap.
- AttributeName: los nombres delimitados por comas de los atributos que se van a indexar. Para los índices de acceso de campo, los nombre de atributo son equivalentes a los nombres de campo. Para los índices de acceso de propiedad, los nombres de atributo son los nombres de propiedad compatibles con JavaBean. Si sólo hay un nombre de atributo, HashIndex es un único índice de atributo y si el atributo es una relación, también es un índice de relación. Si se incluyen varios nombres de atributo en los nombres de atributo, HashIndex es un índice compuesto.
- FieldAccessAttribute: se utiliza para las correlaciones sin entidades. Si tiene el valor true, se accede al objeto utilizando los campos directamente. Si no se especifica o se establece en false, se utiliza el método getter del atributo para acceder a los datos.
- POJOKeyIndex: se utiliza para las correlaciones sin entidad. Si el valor es true, el índice hará una introspección del objeto en la parte de la clave de la correlación. Esto es práctico cuando la clave es una clave compuesta y el valor no tiene que tener ninguna clave incorporada. Si no se especifica o se establece en false, el índice hará una introspección del objeto de la parte del valor de la correlación.
- RangeIndex: si el valor es true, la indexación del rango está habilitada y la aplicación puede difundir el objeto de índice recuperado en la interfaz MapRangeIndex. Si la propiedad RangeIndex está configurada como “false”, la aplicación sólo puede difundir el objeto de índice recuperado en la interfaz MapIndex.

HashIndex de atributo único versus HashIndex compuesto

Si la propiedad AttributeName de HashIndex incluye varios nombres de atributo, HashIndex es un índice compuesto. De lo contrario, si incluye sólo un nombre de atributo, es un índice de atributo único. Por ejemplo, el valor de propiedad AttributeName de un HashIndex compuesto puede ser ″city,state,zipcode”. Incluye tres atributos delimitados por comas. Si el valor de la propiedad AttributeName sólo es “zipcode” que sólo tiene un atributo, es un HashIndex de atributo único.

El HashIndex compuesto proporciona una forma eficaz de buscar objetos almacenados en memoria caché, cuando los criterios de búsqueda implican muchos atributos. Sin embargo, no soporta el índice de rango y su propiedad RangeIndex debe establecerse en “false”.

Consulte el tema sobre un HashIndex compuesto en la *Guía de administración*.

HashIndex de relación

Si el atributo indexado de HashIndex de atributo único es una relación, ya sea con un único valor o con varios, HashIndex es un HashIndex de relación. Para el HashIndex de relación, la propiedad RangeIndex de HashIndex se debe establecer en “false”.

El HashIndex de relación puede acelerar las consultas que utilizan referencias cíclicas o que utilizan los filtros de consulta IS NULL, IS EMPTY, SIZE y MEMBER OF. Consulte la optimización de consulta mediante los índices de la *Guía de programación*.

HashIndex de rango

Si la propiedad RangeIndex de HashIndex se establece en "true", HashIndex es un índice de rango y puede soportar la interfaz MapRangeIndex. Un MapRangeIndex soporta las funciones para buscar los datos utilizando funciones de rango como, por ejemplo mayor que, menor que, o ambos, mientras que un MapIndex sólo soporta las funciones de igual. Para un índice de atributo único, la propiedad RangeIndex se puede establecer en "true" sólo si el atributo indexado es del tipo Comparable. Si el índice de atributo único va a ser utilizado por la consulta, la propiedad RangeIndex se debe establecer en "true" y el atributo indexado debe ser del tipo Comparable. Para el HashIndex de relación y el HashIndex compuesto, la propiedad RangeIndex se debe establecer en "false".

A continuación, aparece un resumen del uso del índice de rango.

Tabla 7. Soporte para el índice de rango

Tipo HashIndex	**Soporta el índice de rango**
HashIndex de atributo único: el atributo indexado o clave es del tipo Comparable	Sí
HashIndex de atributo único: la clave o atributo indexado no es del tipo Comparable	No
Índice compuesto HashIndex	No
HashIndex de relación	No

Optimización de la consulta utilizando HashIndex

La definición y el uso correctos de índices puede mejorar significativamente el rendimiento de las consultas. Las consultas de WebSphere® eXtreme Scale pueden utilizar los plug-ins de HashIndex incorporados para mejorar el rendimiento de las consultas. Aunque el uso de los índices puede mejorar significativamente el rendimiento de la consulta, puede tener un impacto en el rendimiento en las operaciones de correlación transaccional.

Configuración de réplica de igual a igual con JMS

El mecanismo de réplica de igual a igual basada en JMS (Java Message Service) se utiliza en ambos entorno de WebSphere eXtreme Scale, el local y el distribuido. JMS es un proceso de réplica de núcleo a núcleo y permite a las actualizaciones de datos fluir entre los ObjectGrids locales y los ObjectGrids distribuidos. Por ejemplo, con este mecanismo podrá mover actualizaciones de datos de una cuadrícula de eXtreme Scale distribuida a una cuadrícula de eXtreme Scale local, o de una cuadrícula a otra en un dominio de sistema diferente.

Antes de empezar

El mecanismo de réplica de igual a igual basado en JMS se basa en el ObjectGridEventListener basado en JMS incorporado, com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener. Si desea información detallada relacionada con la habilitación del mecanismo de réplica de igual a igual, consulte "Receptor de sucesos JMS" en la página 146.

Si desea más información, consulte "Habilitación del mecanismo de invalidación de clientes" en la página 81.

Lo que aparece a continuación es un ejemplo de configuración XML para habilitar el mecanismo de réplica de igual a igual en una configuración de eXtreme Scale:

```
configuración de réplica de igual a igual - ejemplo de XML<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>
```

Habilitación del mecanismo de invalidación de clientes

En un entorno de WebSphere eXtreme Scale distribuido, el cliente tiene una memoria caché cercana de manera predeterminada al utilizar la estrategia de bloqueo optimista o al inhabilitar el bloqueo. La memoria caché cercana tiene sus propios datos locales almacenados en memoria caché. Si un cliente de eXtreme Scale confirma una actualización, la actualización se produce en el servidor y la memoria caché cercana del cliente. Sin embargo, otros clientes de eXtreme Scale no reciben la información de actualización y podrían tener datos desfasados.

Memoria caché cercana

Las aplicaciones deben estar al tanto del problema de los datos obsoletos en el cliente de eXtreme Scale. Puede utilizar la clase ObjectGridEventListener incorporada basada en el Java Message Service (JMS), com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener, para habilitar el mecanismo de invalidación de cliente dentro de un entorno eXtreme Scale distribuido que es conocido como una cuadrícula de eXtreme Scale.

El mecanismo de invalidación de clientes es la solución al problema de los datos obsoletos de la memoria caché cercana del cliente en el entorno distribuido de eXtreme Scale. Este mecanismo garantiza que la memoria caché cercana del cliente se sincronice con los servidores u otros clientes. Sin embargo, a pesar de este mecanismo de invalidación de clientes basado en JMS, la memoria caché cercana del cliente no se actualiza inmediatamente. Se produce un retardo cuando el tiempo de ejecución de eXtreme Scale publica actualizaciones.

Están disponibles dos modelos para el mecanismo de invalidación de cliente en un entorno de eXtreme Scale distribuido:

- Modelo cliente/servidor: en este modelo, todos los procesos de servidor desempeñan el rol de editor que publica todos los cambios de las transacciones en el destino JMS designado. Todos los procesos de cliente desempeñan los roles de receptor y reciben todos los cambios transaccionales del destino JMS designado.
- Cliente como modelo de roles duales: en este modelo, los procesos de servidor no tienen nada que ver con el destino JMS. Todos los procesos de cliente desempeñan los roles tanto de editor JMS como de receptor. Los cambios transaccionales que se producen en el cliente se publican en el destino JMS y todos los clientes reciben estos cambios transaccionales.

Si desea más información sobre cómo habilitar el mecanismo de invalidación del cliente, consulte “Receptor de sucesos JMS” en la página 146.

Modelo cliente/servidor

En un modelo cliente/servidor, los servidores desempeñan un rol de editor JMS y el cliente desempeña un rol de receptor JMS.

ejemplo de XML del modelo cliente-servidor

```
<?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>
```

Cliente como modelo de roles duales

En un modelo de cliente como roles duales, cada cliente desempeña los roles tanto de editor JMS como de receptor. El cliente publica cada cambio transaccional confirmado en un destino JMS designado y recibe todos los cambios transaccionales confirmados de otros clientes. En este modelo el servidor no tiene nada que ver con JMS.

ejemplo de XML de modelo de roles duales

```
<?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>
```

Distribución de cambios entre máquinas virtuales Java de igual

Los objetos LogSequence y LogElement comunican los cambios que se han producido en una transacción de eXtreme Scale con un plug-in ObjectGridEventListener.

Si desea más información sobre cómo se pueden utilizar los Java Message Service (JMS) para distribuir cambios transaccionales, consulte la información sobre el uso de JMS para distribuir los cambios transaccionales en *Visión general del producto*.

Como requisito previo, ObjectGridManager debe almacenar en memoria caché la instancia ObjectGrid. Si desea más información, consulte los métodos createObjectGrid. El valor booleano de cacheInstance debe establecerse en true.

No es necesario implementar este mecanismo. Existe un mecanismo de réplica de igual a igual incorporado para que utilice esta función. Consulte la información sobre cómo configurar la réplica de igual a igual con JMS en *Guía de administración*.

Los objetos proporcionan una forma sencilla para que una aplicación publique los cambios producidos en un ObjectGrid mediante el transporte de un mensaje a ObjectGrids de igual en máquinas virtuales Java remotas y después los aplique en dicha JVM . La clase LogSequenceTransformer es fundamental para habilitar este

soporte. Este artículo examina cómo escribir un escucha mediante el sistema de mensajería JMS (Java Message Service) para propagar los mensajes. Con ese fin, eXtreme Scale soporta la transmisión de LogSequences que genera una operación de confirmación de la transacción de eXtreme Scale entre varios miembros del clúster de WebSphere Application Server con un plug-in proporcionado por IBM. Esta función no está habilitada de manera predeterminada, pero puede configurarse. Sin embargo, si el consumidor o el productor no es un WebSphere Application Server, podría ser necesario utilizar un sistema de mensajería JMS externo.

Implementación del mecanismo

La clase LogSequenceTransformer y las API ObjectGridEventListener, LogSequence y LogElement permiten el uso de operaciones fiables de publicar y suscribir para distribuir los cambios y filtrar las correlaciones que desea distribuir. Los fragmentos de código de este tema muestran cómo utilizar estas API con JMS para crear un ObjectGrid de igual a igual compartido por aplicaciones que se alojan en un conjunto diverso de plataformas que comparten un transporte de mensajes común.

Inicializar el plug-in

ObjectGrid llama al método initialize del plug-in, parte del contrato de la interfaz ObjectGridEventListener, cuando ObjectGrid se inicia. El método initialize debe obtener sus recursos JMS, incluidos las conexiones, sesiones y editores, e iniciar la hebra que es la escucha JMS.

En los ejemplos siguientes se muestra el método initialize:

ejemplo del método 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();

            // debe iniciarse la conexión para recibir mensajes.
            connection.start();

            // la sesión jms no es transaccional (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);

// iniciar la hebra de la escucha.
            listenerRunning = true;
            listenerThread = new Thread(this);
            listenerThread.start();
        } catch (Throwable e) {
            throw new ObjectGridRuntimeException("Cannot initialize", e);
        }
    }
```

El código para iniciar la hebra utiliza una hebra Java 2 Platform, Standard Edition (Java SE). Si ejecuta un servidor WebSphere Application Server versión 6.x o un servidor WebSphere Application Server versión 5.x Enterprise, utilice la interfaz de programación de aplicaciones (API) del bean asíncrono para iniciar esta hebra de

daemon. También puede utilizar las API comunes. A continuación se muestra un ejemplo de fragmento de código de sustitución que muestra la misma acción mediante el uso de un gestor de trabajo:

```
// iniciar la hebra de la escucha.
listenerRunning = true;
workManager.startWork(this, true);
```

El plug-in también debe implementar la interfaz Work en lugar de la interfaz Runnable. Debe además añadir un método release para establecer la variable listenerRunning en false. El plug-in debe proporcionarse con una instancia WorkManager en su constructor y mediante inyección si se utiliza un contenedor IoC (Inversión de control).

Transmitir los cambios

A continuación se muestra un método transactionEnd de ejemplo para publicar los cambios locales realizados en un ObjectGrid. En este ejemplo se utiliza un JMS, aunque puede utilizarse cualquier transporte de mensajes capaz de producir una mensajería de publicar y suscribir fiable.

ejemplo del método transactionEnd

```
// Este método se sincroniza para garantizar que
    // los mensajes se publican en el orden en que se
    // confirmó la transacción. Si se empieza publicando los mensajes
    // en paralelo, los receptores podrían dañar la correlación
    // ya que las operaciones de supresión pueden llegar antes que las de inserción, etc.
    public synchronized void transactionEnd(String txid, boolean isWriteThroughEnabled, boolean committed,
Collection changes) {
        try {
            // debe utilizarse la modalidad write through y confirmarse.
            if (isWriteThroughEnabled && committed) {
                // escribir las secuencias en un byte []
                ByteArrayOutputStream bos = new ByteArrayOutputStream();
                ObjectOutputStream oos = new ObjectOutputStream(bos);
                if (publishMaps.isEmpty()) {
                    // serializar toda la colección
                    LogSequenceTransformer.serialize(changes, oos, this, mode);
                } else {
                    // filtrar LogSequences basado en el contenido de publishMaps
                    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);
                }
                // realizar un mensaje de objeto para los cambios
                oos.flush();
                ObjectMessage om = jmsSession.createObjectMessage(bos.toByteArray());
                // establecer propiedades
                om.setStringProperty(PROP_TX, txid);
                om.setStringProperty(PROP_GRIDNAME, myGrid.getName());
                // transmitirlo.
                publisher.publish(om);
            }
        } catch (Throwable e) {
            throw new ObjectGridRuntimeException("Cannot push changes", e);
        }
    }
```

Este método utiliza diversas variables de instancia:

- Variable `jmsSession`: sesión JMS que se utiliza para publicar mensajes. Se crea al inicializarse el plug-in.
- Variable `mode`: modo de distribución.
- Variable `publishMaps`: conjunto que contiene el nombre de cada correlación con los cambios que se van a publicar. Si la variable está vacía, se publicarán todas las correlaciones.
- Variable `publisher`: objeto TopicPublisher que se crea durante el método initialize del plug-in.

Recibir y aplicar mensajes de actualización

A continuación se muestra el método run. Este método se ejecuta en un bucle hasta que la aplicación detiene el bucle. Cada repetición del bucle intenta recibir un mensaje JMS y aplicarlo a ObjectGrid.

ejemplo del método de ejecución del mensaje JMS

```
private synchronized boolean isListenerRunning() {
        return listenerRunning;
    }

    public void run() {
        try {
            System.out.println("Listener starting");
            // obtener una sesión jms para recibir los mensajes.
            // No transaccional.
            TopicSession myTopicSession;
            myTopicSession = connection.createTopicSession(false, javax.jms.Session.AUTO_ACKNOWLEDGE);
// obtener un suscriptor para el tema, true indica no recibir
            // mensajes transmitidos mediante editores
            // en esta conexión. De lo contrario, recibiríamos nuestras propias actualizaciones.
            TopicSubscriber subscriber = myTopicSession.createSubscriber(topic, null, true);
System.out.println("Listener started");
            while (isListenerRunning()) {
                ObjectMessage om = (ObjectMessage) subscriber.receive(2000);
                if (om != null) {
                    // Usar objeto Session pasado en el método initialize...
                    // es muy importante utilizarlo, sin write through
                    mySession.beginNoWriteThrough();
                    byte[] raw = (byte[]) om.getObject();
                    ByteArrayInputStream bis = new ByteArrayInputStream(raw);
                    ObjectInputStream ois = new ObjectInputStream(bis);
                    // inflar LogSequences
                    Collection collection = LogSequenceTransformer.inflate(ois, myGrid);
Iterator iter = collection.iterator();
                    while (iter.hasNext()) {
                        // procesar los cambios de las correlaciones de acuerdo con la modalidad
                        // una vez serializado LogSequence
                        LogSequence seq = (LogSequence) iter.next();
                        mySession.processLogSequence(seq);
                    }
                    mySession.commit();
                } // if there was a message
            } // while loop
            // detener la conexión
            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");
    }
```

Receptor de sucesos JMS

JMSObjectGridEventListener se ha diseñado para soportar la invalidación de la memoria caché cercada del cliente y un mecanismo de réplica de igual a igual. Se trata de una implementación JMS (Java Message Service) de la interfaz ObjectGridEventListener.

El mecanismo de invalidación del cliente se puede utilizar en un entorno distribuido de eXtreme Scale para asegurarse de que los datos de la memoria caché cercana del cliente estén sincronizados con los servidores y otros clientes. Sin esta función, la memoria caché cercana del cliente podría albergar datos obsoletos. Sin embargo, incluso con este mecanismo de invalidación de cliente basado en JMS, debe tener en cuenta la ventana de temporización para actualizar una memoria caché cercana debido al retardo para el tiempo de ejecución en la publicación de actualizaciones.

El mecanismo de réplica de igual a igual se puede utilizar en entornos distribuidos y también locales de eXtreme Scale. Se trata de un proceso de réplica de núcleo a

núcleo y permite a las actualizaciones de datos fluir entre los ObjectGrids locales y los ObjectGrids distribuidos. Por ejemplo, con este mecanismo puede mover las actualizaciones de datos de una cuadrícula distribuida a un ObjectGrid local, o de una cuadrícula a otra en un distinto dominio de sistema.

JMSObjectGridEventListener requiere que el usuario configure la información de JMS y JNDI (Java Naming and Directory Interface) para poder obtener los recursos JMS necesarios. De forma adicional, las propiedades relacionadas con la réplica se deben definir de forma correcta. En un entorno JEE, JNDI debe estar disponibles en los contenedores web y también en los contenedores EJB (Enterprise JavaBean). En este caso, la propiedad JNDI es opcional, a menos que desee obtener recursos JMS externos.

Este receptor de sucesos tiene propiedades que puede configurar mediante XML o con enfoques programáticos, que se pueden utilizar sólo para la invalidación de cliente, sólo para la réplica de igual a igual, o ambos. La mayoría de propiedades son opcionales para personalizar el comportamiento para conseguir la funcionalidad necesaria.

.

Si desea más información, consulta la API JMSObjectGridEventListener.

Ampliación del plug-in JMSObjectGridEventListener

El plug-in JMSObjectGridEventListener permite a las instancias de ObjectGrid iguales recibir actualizaciones cuando los datos de la cuadricula se han modificado o desalojado. También permite notificar a los clientes cuando se actualizan o desalojan entradas de una cuadrícula de eXtreme Scale. Este tema describe cómo ampliar el plug-in JMSObjectGridEventListener para permitir notificar a las aplicaciones cuando se recibe un mensaje JMS. Esto es más práctico cuando se utiliza el valor CLIENT_SERVER_MODEL para la invalidación de clientes.

Cuando se ejecuta en el rol de receptor, la ejecución de eXtreme Scale llama automáticamente al método JMSObjectGridEventListener.onMessage alterado temporalmente cuando la instancia de JMSObjectGridEventListener recibe actualizaciones de mensaje JMS de la cuadrícula. Estos mensajes recortan una colección de objetos LogSequence. Los objetos LogSequence se pasan en el método onMessage y la aplicación utiliza el LogSequence para identificar qué entradas de memoria caché se han insertado, suprimido, actualizado o invalidado.

Para utilizar el punto de ampliación onMessage, las aplicaciones realizan los siguientes pasos.

1. Cree una nueva clase, ampliando la clase JMSObjectGridEventListener, alterando temporalmente el método onMessage.
2. Configure el JMSObjectGridEventListener ampliado del mismo modo que el ObjectGridEventListener para ObjectGrid.

El JMSObjectGridEventListener ampliado es una clase hija de JMSObjectGridEventListener y sólo puede alterar temporalmente dos métodos: los métodos initialize (opcional) y onMessage. Si una clase hija de JMSObjectGridEventListener debe utilizar cualquier artefacto de ObjectGrid como, por ejemplo, ObjectGrid o Session en el método onMessage, puede obtener estos artefactos en el método initialize y almacenarlos en la memoria caché como

variables de instancia. Además, en el método onMessage, los artefactos de ObjectGrid almacenados en la memoria caché se pueden utilizar para procesar una colección pasada de LogSequences.

Nota: El método initialize alterado temporalmente debe invocar el método super.initialize para poder inicializar un JMSObjectGridEventListener padre de forma apropiada.

A continuación, aparece un ejemplo de una clase JMSObjectGridEventListener ampliada.

```
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;

    /**
     * Esta es la cuadrícula asociada al receptor.
     */
     ObjectGrid grid;

    /**
     * Esta es la sesión asociada a este receptor.
     */
     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) {
  // Nota: si debe utilizar algún artefacto de ObjectGrid, este clase necesita obtener el ObjectGrid
  //  de la instancia pasada de Session y obtener el ObjectMap de la instancia de la sesión
  // para cualquier operación de correlación de ObjectGrid transaccional.

  super.initialize(session);  // debe invocar el método initialize de super.
  this.session = session; // almacene en la memoria caché la instancia de la sesión, en caso
  //   que necesite utilizarla para realizar la operación de correlación.
  this.grid = session.getObjectGrid(); // obtenga el ObjectGrid, en caso de
  //   necesitar obtener la información de ObjectGrid.

  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 + "]";
 }
}
```

Configuración

La clase JMSObjectGridEventListener ampliada se debe configurar del mismo modo para la invalidación de cliente y, también, para el mecanismo de réplica de igual a igual. Lo que aparece a continuación es el ejemplo de configuración de XML.

```
<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>
```

Nota: El className del bean ObjectGridEventListener se ha configurado con la clase JMSObjectGridEventListener ampliada con las mismas propiedades que el JMSObjectGridEventListener genérico.

Configuración con archivos XML

WebSphere eXtreme Scale se configura mediante una colección de archivos XML. Cada archivo XML sirve a un propósito en la configuración del producto.

Configuración de topología de despliegue

Utilice el XML de descriptor de política de despliegue para gestionar la topología de despliegue.

La política de despliegue está codificada como un archivo XML que se proporciona para un contenedor eXtreme Scale. El archivo XML especifica la siguiente información.

- Las correlaciones que pertenecen a cada conjunto de correlaciones

- El número de particiones
- El número de réplicas síncronas y asíncronas

La política de despliegue también controla los siguientes comportamientos de colocación.

- El número mínimo de contenedores activos antes de que se lleve a cabo la colocación
- Sustitución automática de fragmentos perdidos
- Colocación de cada fragmento desde una sola partición en otra máquina

Si desea más información sobre el archivo XML de la política de despliegue, consulte "Archivo XML de descriptor de la política de despliegue" en la página 152.

La información de punto final no está preconfigurada en el entorno dinámico. En la política de despliegue no hay ningún nombre de servidor ni información de topología física. Todos los fragmentos de una cuadrícula dinámica se colocan automáticamente en contenedores a través del servicio de catálogo. El servicio de catálogo utiliza las restricciones definidas por la política de despliegue para gestionar automáticamente la colocación de fragmentos. Esta colocación de fragmentos automática lleva una fácil configuración para las cuadrículas grandes. También puede añadir servidores al entorno según sea necesario.

Nota: En un entorno WebSphere Application Server, no está soportado un tamaño de grupo principal de más de 50 miembros.

Un archivo XML de política de despliegue se pasa a un contenedor eXtreme Scale durante el arranque. Se debe utilizar una política de despliegue junto con un archivo XML de ObjectGrid. La política de despliegue no es necesaria para iniciar un contenedor, aunque se recomienda. La política de despliegue debe ser compatible con el XML de ObjectGrid que se utiliza con la misma. Para cada elemento objectgridDeployment de la política de despliegue, debe tener un objectGrid correspondiente en el XML de ObjectGrid. Las correlaciones en objectgridDeployment deben ser coherentes con las backingMaps encontradas en el XML de ObjectGrid. Debe hacerse referencia a cada backingMap dentro de únicamente un mapSet.

En el siguiente ejemplo, se intenta emparejar el archivo `companyGridDpReplication.xml` con el correspondiente archivo `companyGrid.xml`.

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>
```

El archivo `companyGridDpReplication.xml` tiene un mapSet dividido en 11 particiones. Cada partición debe tener, exactamente, una réplica síncrona. El número de réplicas síncronas viene dictado por los atributos minSyncReplicas y maxSyncReplicas. Puesto que el atributo minSyncReplicas está establecido en 1, cada partición de mapSet debe tener, como mínimo, una réplica síncrona disponible para procesar transacciones de escritura. Puesto que maxSyncReplicas está establecido en 1, cada partición no debe superar una réplica síncrona. Las particiones de este mapSet no tiene réplicas asíncronas.

El atributo numInitialContainers instruye al servicio de catálogo que difiera la colocación hasta que estén disponibles cuatro contenedores para soportar este ObjectGrid. El atributo numInitialContainers se ignora después de que se haya alcanzado el número especificado de contenedores.

El archivo `companyGridDpReplication.xml` muestra una manera habitual de configurar una política de despliegue, aunque una política de despliegue puede ofrecer incluso más control sobre cómo y cuándo se despliega ObjectGrid en el entorno. Si desea una descripción completa del archivo descriptor de la política de despliegue, consulte “Archivo XML de descriptor de la política de despliegue” en la página 152.

Topología distribuida

Las memorias caché coherentes distribuidas ofrecen un mayor rendimiento, disponibilidad y escalabilidad que puede configurar el usuario.

WebSphere eXtreme Scale equilibra automáticamente los servidores. Puede incluir servidores adicionales sin reiniciar WebSphere eXtreme Scale. La adición de servidores adicionales sin tener que reiniciar eXtreme Scale le permite tener despliegues sencillos y, también, despliegues grandes de terabytes en los que son necesarios cientos de servidores. Esta topología de despliegue es flexible. Con el servicio de catálogo, puede añadir y eliminar servidores para utilizar mejor los recursos sin eliminar toda la memoria caché. Para añadir o eliminar un servidor, utilice el mandato startOgServer para iniciar un servidor de contenedor, que se comunica con el servicio de catálogo con la opción **-catalogServiceEndPoints**. Todos los clientes de la topología distribuida se comunican con el servicio de catálogo a través del protocolo IIOP (Internet Interoperability Object Protocol). Todos los clientes utilizan la interfaz de ObjectGrid para comunicarse con los servidores.

La prestación de la configuración dinámica de WebSphere eXtreme Scale facilita la adición de recursos al sistema. Los contenedores alojan los datos y las funciones de servicio de catálogo como punto táctil para la cuadrícula. Los contenedores son responsables del mantenimiento de datos. El servicio de catálogo es responsable del reenvío de solicitudes al lugar correcto la primera vez, asignando espacio de contenedores de host, y gestionando el estado y la disponibilidad del sistema en general. Los clientes se conectan a un servicio de catálogo, recuperan una descripción de la topología de contenedor-servidor, y luego se comunican directamente con cada servidor cuando sea necesario. Cuando la topología del

servidor cambia debido a la adición de nuevos servidores, o debido a la anomalía de otros, el se direcciona automáticamente al servidor apropiado que aloja los datos.

Normalmente, un servicio de catálogo existe en su propia cuadrícula de Máquinas virtuales Java . Se puede utilizar un único servicio de catálogo para gestionar diversos servidores. Un contenedor se puede iniciar en una JVM por sí mismo o puede cargar en una JVM arbitraria con otros contenedores para distintos servidores. Un cliente puede existir en cualquier JVM y comunicarse con uno o más servidores. También puede existir un cliente en la misma JVM que la de un contenedor.

Archivo XML de descriptor de la política de despliegue

Para configurar una política de despliegue, utilice un archivo XML de descriptor de política de despliegue.

En las siguientes secciones, se definen los elementos y atributos del archivo XML de descriptor de la política de despliegue. Consulte "Archivo `deploymentPolicy.xsd`" en la página 156 si desea ver un ejemplo del esquema XML de la política de despliegue.

Elemento deploymentPolicy

El elemento deploymentPolicy es el elemento de nivel superior del archivo XML de política de despliegue. Este elemento configura el espacio de nombres del archivo y la ubicación del esquema. El esquema se define en el archivo `deploymentPolicy.xsd`.

- Número de apariciones: una
- Elemento hijo: elemento objectgridDeployment

Elemento objectgridDeployment

El elemento objectgridDeployment se utiliza para hacer referencia a una instancia de ObjectGrid desde el archivo XML de ObjectGrid. Dentro del elemento objectgridDeployment puede dividir las correlaciones en conjuntos de correlaciones.

- Número de apariciones: una o más
- Elemento hijo: elemento mapSet

Atributos

objectgridName
Especifica el nombre de el ObjectGrid a desplegar. Este atributo hace referencia a un elemento objectGrid que está definido en el archivo XML de ObjectGrid. (Necesario)

```
<objectgridDeployment objectgridName="objectgridName"/>
```

Por ejemplo, el atributo objectgridName se ha establecido como `CompanyGrid` en el archivo `companyGridDpReplication.xml`. El atributo objectgridName hace referencia a `CompanyGrid` que se ha definido en el archivo `companyGrid.xml`. Para obtener más información, consulte "Archivo XML de descriptor ObjectGrid" en la página 157.

Elemento mapSet

El elemento mapSet se utiliza para agrupar correlaciones. Las correlaciones dentro de un elemento mapSet se particionan y replican de forma parecida. Cada correlación debe pertenecer sólo a un mapSet.

- Número de apariciones: una o más
- Elemento hijo: elemento map

Atributos

name
Especifica el nombre del mapSet. Este atributo debe ser exclusivo dentro del elemento objectgridDeployment. (Necesario)

numberOfPartitions
Especifica el número de particiones para el mapSet. El valor predeterminado es 1. El número debe ser adecuado para el número de contenedores que alojan las particiones. (Opcional)

minSyncReplicas
Especifica el número mínimo de réplicas síncronas para cada partición en el mapSet. El valor predeterminado es 0. Los fragmentos no se colocan hasta que el dominio pueda dar soporte al número mínimo de réplicas síncronas. Para dar soporte al valor minSyncReplicas, es necesario un contenedor más que el valor de minSyncReplicas. Si el número de réplicas síncronas cae por debajo del valor de minSyncReplicas, ya no se permiten transacciones de grabación para esa partición. (Opcional)

maxSyncReplicas
Especifica el número máximo de réplicas síncronas para cada partición en el mapSet. El valor predeterminado es 0. No se colocan más réplicas síncronas para una partición una vez que un dominio alcanza este número de réplicas síncronas para esa partición específica. La adición de contenedores que puedan dar soporte a este ObjectGrid puede comportar un aumento en el número de réplicas síncronas si todavía no se ha alcanzado el valor de maxSyncReplicas. (Opcional)

maxAsyncReplicas
Especifica el número máximo de réplicas asíncronas para cada partición en el mapSet. El valor predeterminado es 0. Una vez que se han colocado el primario y todas las réplicas síncronas para una partición, las réplicas asíncronas se colocan hasta que se alcanza el valor de maxAsyncReplicas. (Opcional)

replicaReadEnabled
Si este atributo se establece en `true`, las solicitudes de lectura se distribuyen entre un primario de la partición y sus réplicas. Si el atributo replicaReadEnabled es `false`, las solicitudes de lectura se direccionan únicamente al primario. El valor predeterminado es false. (Opcional)

numInitialContainers
Especifica el número de contenedores de eXtreme Scale que son necesarios antes de que se produzca la colocación inicial para los fragmentos de este mapSet. El valor predeterminado es 1. Este atributo puede ayudar a ahorrar ancho de banda de la red y CPU cuando se pone en línea un ObjectGrid. (Opcional)

Al iniciar un contenedor eXtreme Scale se envía un suceso al servicio de catálogo. La primera vez que el número de contenedores activos es igual al

valor numInitialContainers de un mapSet, el servicio de catálogo coloca los fragmentos del mapSet, siempre que también se pueda satisfacer al valor de minSyncReplicas. Una vez que se ha alcanzado el valor de numInitialContainers, cada suceso iniciado por contenedor puede desencadenar que se vuelvan a equilibrar fragmentos sin colocar y colocados anteriormente. Si sabe la cantidad aproximada de contenedores que se van a iniciar para este mapSet, puede establecer el valor de numInitialContainers en un valor cercano a ese número para evitar que se vuelva a equilibrar después del inicio de cada contenedor. La colocación no se lleva a cabo hasta que se alcanza el valor de numInitialContainers para el mapSet.

autoReplaceLostShards
Especifica si los fragmentos perdidos se colocan en otros contenedores. El valor predeterminado es true. Cuando un contenedor se detiene o sufre una anomalía, se pierden los fragmentos que se ejecutan en el mismo. Un primario perdido asciende a una de sus réplicas para que se convierta en el nuevo primario. Debido a este ascenso, se pierde una de las réplicas. En determinados casos, puede que no desee que los fragmentos perdidos se sustituyan automáticamente en contenedores disponibles. Si desea que los fragmentos perdidos permanezcan sin colocar, establezca el atributo autoReplaceLostShards en `false`. Este valor no afecta a la cadena de ascensos, sólo la sustitución del último fragmento en la cadena. (Opcional)

developmentMode
Con este atributo, puede influir en el lugar en el que se coloca el fragmento en relación a sus fragmentos iguales. El valor predeterminado es true. Cuando el atributo developmentMode se establece en `false`, ninguno de los dos fragmentos de la misma partición se colocan en la misma máquina. Cuando el atributo developmentMode se establece en `true`, los fragmentos de la misma partición se pueden colocar en la misma máquina. En cualquier caso, ninguno de los dos fragmentos de la misma partición pueden colocarse en el mismo contenedor. (Opcional)

placementStrategy
Existen dos estrategias de colocación. La estrategia predeterminada es `FIXED_PARTITION`, donde el número de fragmentos primarios que se colocan entre los contenedores disponibles es igual al número de particiones definidas, aumentadas por el número de replicas. La estrategia alternativa es `PER_CONTAINER`, donde el número de fragmentos primarios que se colocan en cada contenedor es igual al número de particiones que se han definido, con un número igual de réplicas colocadas en otros contenedores. (Opcional)

```
<mapSet
(1)   name="mapSetName"
(2)   numberOfPartitions="numberOfPartitions"
(3)   minSyncReplicas="minimumNumber"
(4)    maxSyncReplicas="maximumNumber"
(5)   maxAsyncReplicas="maximumNumber"
(6)   replicaReadEnable="true" | "false"
(7)   numInitialContainers="numberOfInitialContainersBeforePlacement"
(8)   autoReplaceLostShards="true" | "false"
(9)   developmentMode="true" | "false"
(10)   placementStrategy="FIXED_PARTITION"|"PER_CONTAINER"
/>
```

En el siguiente ejemplo, se utiliza el elemento mapSet para configurar una política de despliegue. El valor se establece en `mapSet1` y está dividido en 10 particiones. Cada una de estas particiones debe tener como mínimo una réplica síncrona disponible y no más de dos réplicas síncronas. Cada partición también tiene una réplica síncrona si el entorno puede darle soporte. Se colocan todas las réplicas síncronas antes de que se coloquen las réplicas asíncronas. Además, el servicio de catálogo no intenta colocar los fragmentos para mapSet1 hasta que el dominio

pueda dar soporte al valor minSyncReplicas. El soporte del valor de minSyncReplicas requiere dos o más contenedores: uno para el primario y dos para la réplica síncrona.

```
<?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>
```

Aunque sólo son necesarios dos contenedores para satisfacer los valores de duplicación, el atributo numInitialContainers requiere 10 contenedores disponibles antes de que el servicio de catálogo intente colocar cualquiera de los fragmentos en este mapSet. Una vez que el dominio tiene 10 contenedores que pueden dar soporte a CompanyGrid ObjectGrid, se colocan todos los fragmentos de mapSet1.

Como el atributo autoReplaceLostShards está establecido en true, cualquier fragmento de este mapSet que se pierde como resultado de la anomalía del contenedor se sustituye automáticamente en otro contenedor, siempre que haya un contenedor disponible para alojar el fragmento perdido. Los fragmentos de la misma partición no pueden colocarse en la misma máquina para mapSet1 porque el atributo developmentMode está establecido en `false`. Las solicitudes de sólo lectura se distribuyen a lo largo del primario y sus réplicas para cada partición porque el valor replicaReadEnabled es `true`.

Elemento map

Cada correlación de un elemento mapSet hace referencia a los elementos backingMap que se definen en el archivo XML de ObjectGrid. Cada correlación de un eXtreme Scale distribuido debe pertenecer sólo a un elemento de mapSet. Consulte “Archivo `objectGrid.xsd`” en la página 174 si desea más información sobre el archivo XML de ObjectGrid.

- Número de apariciones: una o más
- Elemento hijo: ninguno

Atributos

ref

Proporciona una referencia a un elemento backingMap en el archivo XML de ObjectGrid. Cada correlación de un mapSet debe hacer referencia a una backingMap del archivo XML de ObjectGrid. El valor que se asigna al atributo ref debe coincidir con el atributo name de uno de los elementos backingMap del archivo XML de ObjectGrid. (Necesario)

```
<map
(1)   ref="backingMapReference"
/>
```

El archivo `companyGridDpMapSetAttr.xml` utiliza el atributo ref en la correlación para hacer referencia a cada una de las backingMaps del archivo `companyGrid.xml`.

Para obtener información sobre cómo evitar conflictos en la configuración XML, consulte “Resolución de problemas de configuración de XML” en la página 86.

Archivo `deploymentPolicy.xsd`:

Utilice el esquema XML de la política de despliegue para crear un archivo XML del descriptor de despliegue.

Consulte “Archivo XML de descriptor de la política de despliegue” en la página 152 si desea descripciones de los elementos y atributos definidos en el archivo `deploymentPolicy.xsd`.

```
<?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>
```

Archivo XML de descriptor ObjectGrid

Para configurar WebSphere eXtreme Scale, utilice el archivo XML de descriptor de ObjectGrid y la API ObjectGrid.

En las siguientes secciones, se proporcionan archivos XML de ejemplo para mostrar varias configuraciones. Cada elemento y atributo del archivo XML está definido. Utilice el esquema XML de descriptor de ObjectGrid para crear el archivo XML de descriptor. Consulte “Archivo objectGrid.xsd” en la página 174 si desea ver un ejemplo del esquema XML de descriptor de ObjectGrid.

Se utiliza una versión modificada del archivo companyGrid.xml original. El siguiente archivo companyGridSingleMap.xml es parecido al archivo companyGrid.xml. El archivo companyGridSingleMap.xml tiene una correlación y el archivo companyGrid.xml tiene cuatro correlaciones. Los elementos y atributos del archivo se describen en detalle en el siguiente ejemplo.

```
<?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>
```

Elemento objectGridConfig

El elemento objectGridConfig es el elemento de nivel superior del archivo XML. Grabe este elemento en el documento XML de eXtreme Scale tal como se muestra en el ejemplo anterior. Este elemento configura el espacio de nombres del archivo y la ubicación del esquema. El esquema se define en el archivo `objectGrid.xsd`.

- Número de apariciones: una
- Elemento hijo: elemento objectGrids y elemento backingMapPluginCollections

Elemento objectGrids

El elemento objectGrids es un contenedor para todos los elementos objectGrid. En el archivo `companyGridSingleMap.xml`, el elemento objectGrids contiene un ObjectGrid, la objectGrid CompanyGrid.

- Número de apariciones: una o más
- Elemento hijo: elemento objectGrid

Elemento objectGrid

Utilice el elemento objectGrid para definir un ObjectGrid. Cada uno de los atributos del elemento objectGrid corresponde a un método en la interfaz ObjectGrid.

- Número de apariciones: una a muchas
- Elemento hijo: elemento bean, elemento backingMap, elemento querySchema y elemento streamQuerySet

Atributos

name
Especifica el nombre que se asigna a ObjectGrid. La validación XML falla si falta este atributo. (Necesario)

securityEnabled
Habilita la seguridad en el nivel ObjectGrid, que habilita las autorizaciones de acceso a los datos de la correlación, cuando establece el atributo en `true`. El valor predeterminado es `true`. (Opcional)

authorizationMechanism
Establece el mecanismo de autorización para el elemento. Puede establecer el atributo en uno de estos dos valores: `AUTHORIZATION_MECHANISM_JAAS` o `AUTHORIZATION_MECHANISM_CUSTOM`. El valor predeterminado es `AUTHORIZATION_MECHANISM_JAAS`. Establézcalo en `AUTHORIZATION_MECHANISM_CUSTOM` cuando utilice un plug-in MapAuthorization personalizado. Debe establecer el atributo securityEnabled en `true` para que el atributo authorizationMechanism entre en vigor. (Opcional)

permissionCheckPeriod
Especifica un valor entero en segundos que indica la frecuencia con la que se ha de comprobar el permiso que se utiliza para permitir un acceso de cliente. El valor predeterminado es `0`. Cuando se establece el valor de atributo `0`, cada llamada al método get, put, update, remove o evict solicita al mecanismo de autorización, ya sea la autorización JAAS (Java Authentication and Authorization Service) o a la autorización personalizada, que compruebe si el

asunto actual tiene permiso. Un valor mayor que 0 indica el número de segundos que tarda en copiar en caché un conjunto de permisos antes de volver al mecanismo de autorización para que los renueve. Debe establecer el atributo securityEnabled en `true` para que el atributo permissionCheckPeriod entre en vigor. (Opcional)

txTimeout
Especifica la cantidad de tiempo en segundos que se permite la finalización de una transacción. Si una transacción no finaliza en esta cantidad de tiempo, la transacción se marca para la retrotracción y se genera una excepción TransactionTimeoutException. Si establece el valor `0`, las transacciones nunca exceden el tiempo de espera. (Opcional)

entityMetadataXMLFile
Especifica la vía de acceso relativa al archivo XML de descriptor de entidad. La vía de acceso es relativa a la ubicación del archivo descriptor de Objectgrid. Utilice este atributo para definir un esquema de entidad utilizando un archivo XML. Las entidades deben definirse antes de iniciar eXtreme Scale para que cada entidad pueda enlazarse con una BackingMap. (Opcional)

```
<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"
/>
```

En el siguiente ejemplo, el archivo `companyGridObjectGridAttr.xml` muestra una forma de configurar los atributos de un elemento objectGrid. La seguridad está habilitada, el mecanismo de autorización se establece en JAAS y el periodo de comprobación de permisos en 45 segundos. El archivo también registra entidades especificando un atributo entityMetadataXMLFile.

```
<?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>
```

El siguiente código de ejemplo muestra el enfoque programático para lograr la misma configuración que el archivo `companyGridObjectGridAttr.xml` en el ejemplo anterior.

```
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"));
```

Elemento backingMap

El elemento backingMap se utiliza para definir una instancia BackingMap de un ObjectGrid. Cada uno de los atributos del elemento backingMap corresponde a un método de la interfaz BackingMap. Si desea detalles, consulte “Interfaz BackingMap” en la página 67.

- Número de apariciones: cero a muchas
- Elemento hijo: elemento timeBasedDBUpdate

Atributos

name
Especifica el nombre que se asigna a la instancia backingMap. Si falta este atributo, la validación XML fallará. (Necesario)

readOnly
Establece una instancia BackingMap como lectura-grabación cuando se especifica el atributo como `false`. Cuando se especifica el atributo como `true`, la instancia BackingMap es de sólo lectura. La llamada a Session.getMap(String) generará una creación de correlación dinámica si el nombre pasado en el método coincide con la expresión regular especificada en el atributo de nombre de este backingMap. El valor predeterminado es `false`. (Opcional)

plantilla
Especifica si se pueden utilizar las correlaciones dinámicas. Establezca este valor en `true` si la correlación BackingMap es una correlación de plantilla. Las correlaciones de plantilla se pueden utilizar para crear dinámicamente correlaciones cuando se inicia ObjectGrid. Las llamadas a Session.getMap(String) generan que se creen correlaciones dinámicas si el nombre pasado en el método coincide con la expresión regular especificada en el atributo de nombre de backingMap. El valor predeterminado es `false`. (Opcional)

pluginCollectionRef
Especifica una referencia al plug-in backingMapPluginCollection. El valor de este atributo debe coincidir con el atributo de ID de un plug-in backingMapCollection. La validación falla si no existe ningún ID coincidente. Establezca el atributo de forma que se vuelvan a utilizar los plug-ins BackingMap. (Opcional)

numberOfBuckets
Especifica el número de grupos para la instancia BackingMap que se debe utilizar. La instancia BackingMap utiliza una correlación hash para la implementación. Si existen varias entradas en la BackingMap, si dispone de más grupos se obtendrá un mejor rendimiento porque el riesgo de colisiones disminuye a medida que aumenta el número de grupos. Un mayor número de grupos también implica mayor simultaneidad. Especifique un valor de `0` para inhabilitar la memoria caché cercana de un cliente cuando se está comunicando de forma remota con eXtreme Scale. Cuando establece el valor en `0` para un cliente, establezca el valor únicamente en el archivo de descriptor XML de ObjectGrid de alteración temporal de cliente. (Opcional)

preloadMode
Establece la modalidad de precarga si el plug-in de cargador se establece para esta instancia de BackingMap. El valor predeterminado es `false`. Si el atributo se establece en `true`, de forma asíncrona se invoca el método Loader.preloadMap(Session, BackingMap). De lo contrario, la ejecución del método se bloquea al cargar datos de modo que la memoria caché no está disponible hasta que la precarga finaliza. La precarga se produce durante la inicialización. (Opcional)

lockStrategy
Especifica si el gestor de bloqueo interno se utiliza cuando una transacción

acceder a una entrada de correlación. Establezca este atributo en uno de tres valores: `OPTIMISTIC`, `PESSIMISTIC` o `NONE`. El valor predeterminado es `OPTIMISTIC`. (Opcional)

Normalmente, la estrategia de bloqueo optimista se utiliza cuando una correlación no tiene un plug-in de cargador, la correlación en su mayor parte se lee y no se graba en ella ni se actualiza con frecuencia, y el bloqueo no lo proporciona el gestor de persistencia utilizando eXtreme Scale como una memoria caché secundaria, ni la aplicación. Un bloqueo exclusivo se obtiene en una entrada de correlación que se inserta, actualiza o elimina durante la confirmación. El bloqueo garantiza que otra transacción no pueda cambiar la información de versión mientras la transacción que se confirma realice una comprobación de versión optimista.

La estrategia de bloqueo pesimista normalmente se usa para una correlación que no tiene un plug-in de cargador y el bloqueo no está proporcionado por el gestor de persistencia utilizando un eXtreme Scale como memoria caché lateral, por el plug-in de cargador o por la aplicación. La estrategia de bloqueo pesimista se utiliza cuando la estrategia de bloqueo optimista falla con demasiada frecuencia debido a que las transacciones de actualización colisionan frecuentemente en la misma entrada de correlación.

La estrategia sin bloqueo indica que el LockManager interno no es necesario. El control de simultaneidad se proporciona fuera de eXtreme Scale, ya sea a través del gestor de persistencia que utiliza eXtreme Scale como una memoria caché secundaria o aplicación, o mediante el plug-in de cargador que utiliza los bloqueos de base de datos para controlar la concurrencia.

Si desea más información consulte “Bloqueo de entrada de correlación” en la página 71.

numberOfLockBuckets
Establece el número de grupos de bloqueo que utiliza el gestor de bloqueos para la instancia de BackingMap. Establezca el atributo lockStrategy en `OPTIMISTIC` o `PESSIMISTIC` para crear un gestor de bloqueos para la instancia de BackingMap. El gestor de bloqueos utiliza una correlación hash para realizar un seguimiento de las entradas bloqueadas por una o más transacciones. Si existen muchas entradas, si dispone más grupos de bloqueo se obtendrá un mejor rendimiento porque el riesgo de colisiones disminuye a medida que aumenta el número de grupos. Un número mayor de grupos también implica mayor simultaneidad. Establezca el atributo lockStrategy en `NONE` para especificar que la instancia de BackingMap no utilice el gestor de bloqueos. (Opcional)

lockTimeout
Establece el tiempo de espera de bloqueo que utiliza el gestor de bloqueos para la instancia de BackingMap. Establezca el atributo lockStrategy en `OPTIMISTIC` o `PESSIMISTIC` para crear un gestor de bloqueos para la instancia de BackingMap. Para impedir que se produzcan puntos muertos, el gestor de bloqueos tiene un valor de tiempo de espera predeterminado de 15 segundos. Si se supera el límite de tiempo de espera, se produce una excepción LockTimeoutException. El valor predeterminado de 15 segundos es suficiente para la mayoría de las aplicaciones, pero en un sistema con mucha cara, el tiempo de espera puede producirse cuando no existe ningún punto muerto. Utilice el atributo lockTimeout para aumentar el valor del valor predeterminado para impedir que se produzcan excepciones de tiempo de espera excedido falsas. Establezca el atributo lockStrategy en `NONE` para especificar que la instancia de BackingMap no utilice el gestor de bloqueos. (Opcional)

CopyMode
Especifica si una operación get de una entrada de la instancia de BackingMap devuelve el valor real, una copia del valor o un proxy para el valor. Establezca el atributo CopyMode en uno de estos cinco valores:

COPY_ON_READ_AND_COMMIT
El valor predeterminado es COPY_ON_READ_AND_COMMIT. Establezca el valor en COPY_ON_READ_AND_COMMIT para asegurar que una aplicación nunca tenga una referencia a un objeto de valor que esté en la instancia de BackingMap. En cambio, la aplicación siempre funciona con una copia del valor que está en la instancia de BackingMap. (Opcional)

COPY_ON_READ
Establezca el valor en COPY_ON_READ para mejorar el rendimiento sobre el valor de COPY_ON_READ_AND_COMMIT eliminando la copia que se produce cuando se confirma una transacción. Para conservar la integridad de los datos de BackingMap, la aplicación confirma la supresión de cada referencia a una entrada una vez que la transacción se ha confirmado. Si se establece este valor hace que un método ObjectMap.get devuelva una copia del valor en lugar de una referencia al valor, lo que garantiza que los cambios que la aplicación realice en el valor no afecten al elemento BackingMap hasta que la transacción se haya confirmado.

COPY_ON_WRITE
Establezca el valor en COPY_ON_WRITE para mejorar el rendimiento sobre el valor COPY_ON_READ_AND_COMMIT eliminando la copia que se produce la primera vez que una transacción de una clave dada llama al método ObjectMap.get. En cambio, el método ObjectMap.get devuelve un proxy al valor en lugar de una referencia directa al objeto de valor. El proxy garantiza que no se haga una copia del valor salvo que la aplicación llame a un método set en la interfaz de valor.

NO_COPY
Establezca el valor en NO_COPY para permitir que una aplicación nunca modifique un objeto de valor que se haya obtenido utilizando un método ObjectMap.get a cambio de mejoras de rendimiento. Establezca el valor en NO_COPY para las correlaciones asociadas a las entidades de la API de EntityManager.

COPY_TO_BYTES
Establezca el valor en COPY_TO_BYTES para mejorar la huella de la memoria para los tipos Object complejos y para mejorar el rendimiento cuando la copia de un Object se basa en la serialización para realizar la copia. Si un Object no se puede clonar o no se proporciona un ObjectTransformer personalizado con un método copyValue eficaz, el mecanismo de copia predeterminado es serializar e inflar el objeto para realizar una copia. Con el valor COPY_TO_BYTES, la operación de inflar sólo se realiza durante la lectura y la operación de serialización sólo se realiza durante el compromiso.

Si desea más información sobre estos valores, consulte la información sobre los procedimientos recomendados de CopyMode en *Guía de programación*..

valueInterfaceClassName
Especifica una clase que es necesaria al establecer el atributo CopyMode en COPY_ON_WRITE. Este atributo se ignora para las demás modalidades. El valor COPY_ON_WRITE utiliza un proxy cuando se realizan llamadas al método

ObjectMap.get. El proxy garantiza que no se haga una copia del valor salvo que la aplicación llame a un método set en la clase que se especifica como atributo valueInterfaceClassName. (Opcional)

copyKey
Especifica si la copia de la clave es necesaria cuando se crea una entrada de correlación. Si se copia el objeto de clave se permite que la aplicación utilice el mismo objeto de clave para cada operación ObjectMap. Establezca el valor en `true` para copiar el objeto de clave cuando se crea una entrada de correlación. El valor predeterminado es `false`. (Opcional)

nullValuesSupported
Establezca el valor en `true` para dar soporte a valores nulos en la ObjectMap. Cuando se da soporte a valores nulos, una operación get que devuelve un nulo podría significar que el valor es nulo o que la correlación no contiene la clave que se ha pasado al método. El valor predeterminado es `true`. (Opcional)

ttlEvictorType
Especifica cómo se calcula la hora de caducidad de una entrada BackingMap. Establezca este atributo en uno de tres valores: `CREATION_TIME`, `LAST_ACCESS_TIME` o `NONE`. El valor`CREATION_TIME` indica que la hora de caducidad de una entrada es la suma de la hora de creación de la entrada más el valor del atributo timeToLive. El valor `LAST_ACCESS_TIME` indica que la hora de caducidad de una entrada es la suma de la hora del último acceso de la entrada más el valor del atributo timeToLive. El valor `NONE`, que es el valor predeterminado, indica que una entrada no tiene hora de caducidad y que estará presente en la instancia de BackingMap hasta que la aplicación elimine o anule de forma explícita la entrada. (Opcional)

timeToLive
Especifica en segundos cuánto tiempo existe cada entrada de correlación. El valor predeterminado `0` significa que la entrada de correlación siempre existe, o hasta que la aplicación elimina o anula la entrada de forma explícita. De lo contrario, el desalojador TTL desaloja la entrada de correlación basándose en este valor. (Opcional)

streamRef
Especifica que la backingMap es una correlación de origen de corriente. Cualquier inserción o actualización a la backingMap se convierte en un suceso de modalidad continua para el motor de consulta de secuencia. Este atributo debe hacer referencia a un nombre de secuencia válido dentro de un streamQuerySet. (Opcional)

viewRef
Especifica que la backingMap es una correlación de vistas. La salida de vista del motor de consulta de secuencia se convierte en formato tuple de eXtreme Scale y se coloca en la correlación. (Opcional)

evictionTriggers
Establece los tipos de desencadenantes de desalojo adicionales que se deben usar. Todos los desalojos de la correlación de respaldo utilizan esta lista de desencadenantes adicionales. Para evitar una IllegalStateException, se debe invocar este atributo antes de invocar el método ObjectGrid.initialize(). Además, tenga en cuenta que el método ObjectGrid.getSession() invoca de forma implícita el método ObjectGrid.initialize() si la aplicación todavía no ha invocado el método. Las entradas de la lista de desencadenantes están separadas por signos de punto y coma. Los desencadenadores del desalojo Current incluyen MEMORY_USAGE_THRESHOLD. (Opcional)

```
<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|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"
/>
```

En el siguiente ejemplo, el archivo companyGridBackingMapAttr.xml se utiliza para mostrar una configuración de backingMap de ejemplo.

```
<?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>
```

El siguiente código de ejemplo demuestra el enfoque programático para obtener la misma configuración que el archivo companyGridBackingMapAttr.xml en el ejemplo anterior:

```
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);

// cuando se establece la modalidad de copia en COPY_ON_WRITE, es necesaria la clase valueInterface
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); // establecer el tiempo de vida en 50 minutos
```

Elemento bean

Utilice el elemento bean para definir plug-ins. Puede conectar plug-ins a instancias de los elementos objectGrid y BackingMap.

- Número de apariciones dentro del elemento objectGrid: cero a muchas
- Número de apariciones dentro del elemento backingMapPluginCollection: cero a muchas

- Elemento hijo: elemento property

Atributos

id Especifica el tipo de plug-in que se va a crear. (Necesario)

En la siguiente lista se incluyen los plug-ins válidos para un bean que es un elemento hijo del elemento objectGrid:

- Plug-in TransactionCallback
- Plug-in ObjectGridEventListener
- Plug-in SubjectSource
- Plug-in MapAuthorization
- Plug-in SubjectValidation

En la siguiente lista se incluyen los plug-ins válidos para un bean que es un elemento hijo del elemento backingMapPluginCollection:

- Plug-in Loader
- Plug-in ObjectTransformer
- Plug-in OptimisticCallback
- Plug-in Evictor
- Plug-in MapEventListener
- Plug-in MapIndex

className
Especifica el nombre de la clase o bean de spring de la que se debe crear una instancia para crear el plug-in. La clase debe implementar la interfaz de tipo plug-in. Por ejemplo, si especifica `ObjectGridEventListener` como valor para el atributo id, el valor del atributo className debe hacer referencia a una clase que implemente la interfaz ObjectGridEventListener. (Necesario)

```
<bean
(1)   id="TransactionCallback" | "ObjectGridEventListener" |"SubjectSource" |
    "MapAuthorization" | "SubjectValidation" | "Loader" | "ObjectTransformer" |
    "OptimisticCallback" | "Evictor" | "MapEventListener" | "MapIndexPlugin"
(2)   className="class name" | "(spring)bean name"
/>
```

En el siguiente ejemplo el archivo `companyGridBean.xml` se utiliza para mostrar cómo configurar plug-ins utilizando el elemento bean. Se añade un plug-in ObjectGridEventListener al ObjectGrid CompanyGrid. El atributo className para este bean es la clase com.ibm.websphere.objectgrid.plugins.builtins.TranPropListener. Esta clase implementa la interfaz com.ibm.websphere.objectgrid.plugins.ObjectGridEventListener según sea necesario.

En el archivo `companyGridBean.xml` también se define un plug-in BackingMap. Un plug-in evictor se añade a la instancia de BackingMap Customer. Dado que el ID de bean es Evictor, el atributo className debe especificar una clase que implemente la interfaz com.ibm.websphere.objectgrid.plugins.Evictor. La clase com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor implementa esta interfaz. La backingMap hace referencia a sus plug-ins utilizando el atributo 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>
```

El siguiente código de ejemplo muestra el enfoque programático para lograr la misma configuración que el archivo `companyGridBean.xml` en el ejemplo anterior.

```
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);
```

Elemento property

Utilice el elemento property para añadir propiedades a plug-ins. El nombre de la propiedad debe corresponder a un método set de la clase a la que hace referencia el bean que lo contiene.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

name
Especifica el nombre de la propiedad. El valor que se asigna a este atributo debe corresponder a un método set de la clase que se proporciona como el atributo className de bean que lo contiene. Por ejemplo, si establece el atributo className del bean en `com.ibm.MyPlugin` y el nombre de la propiedad que se proporciona es `size`, la clase `com.ibm.MyPlugin` debe tener un método setSize. (Necesario)

type
Especifica el tipo de la propiedad. El tipo se pasa al método set identificado por el atributo name. Los valores válidos son los primitivos Java, sus correspondientes java.lang y java.lang.String. Los atributos de nombre y tipo deben corresponder a una signatura de método del atributo className del bean. Por ejemplo, si establece el nombre como `size` y el tipo como `int`, debe haber un método setSize(int) en la clase que se especifica como el atributo className para el bean. (Necesario)

value
Especifica el valor de la propiedad. Este valor se convierte en el tipo que se especifica mediante el atributo de tipo y se utiliza como un parámetro en la llamada al método set que se identifica mediante los atributos de nombre y tipo. El valor de este atributo no se valida de ningún modo. (Necesario)

description
Describe la propiedad. (Opcional)

```
<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"
/>
```

En el siguiente ejemplo, el archivo companyGridProperty.xml se utiliza para mostrar cómo añadir un elemento property a un bean. En este ejemplo, un elemento property con el nombre maxSize y el tipo int se añade a un desalojador. El desalojador com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor tiene una firma de método que coincide con el método setMaxSize(int). El valor entero 499 se pasa al método setMaxSize(int) en la clase com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor.

```
<?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="Tamaño máximo del desalojador LRU"/>
   </bean>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

El siguiente código de ejemplo muestra el enfoque programático para lograr la misma configuración que el archivo companyGridProperty.xml en el ejemplo anterior.

```
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();
// si en su lugar se utiliza el archivo XML,
// la propiedad añadida provocará que se realice la siguiente llamada
lruEvictor.setMaxSize(449);
customerMap.setEvictor(lruEvictor);
```

Elemento backingMapPluginsCollections

El elemento backingMapPluginsCollections es un contenedor para todos los elementos backingMapPluginCollection. En el archivo companyGridProperty.xml de la sección anterior, el elemento backingMapPluginCollections contiene un elemento backingMapPluginCollection con el ID customerPlugins.

- Número de apariciones: cero a ninguna
- Elemento hijo: elemento backingMapPluginCollection

Elemento backingMapPluginCollection

El elemento backingMapPluginCollection define los plug-ins de BackingMap y se identifica mediante el atributo **id**. Especifique el atributo pluginCollectionRef para

hacer referencia a los plug-ins. Al configurar varios plug-ins BackingMaps de forma parecida, cada BackingMap puede hacer referencia al mismo elemento backingMapPluginCollection.

- Número de apariciones: cero a muchas
- Elemento hijo: elemento bean

Atributos

id Identifica la backingMapPluginCollection, a la que hace referencia el atributo pluginCollectionRef del elemento backingMap. Cada ID debe ser exclusivo. Si el valor de un atributo pluginCollectionRef no coincide con el ID de un elemento backingMapPluginCollection, la validación XML falla. Cualquier número de elementos backingMap pueden hacer referencia a un solo elemento backingMapPluginCollection. (Necesario)

```
<backingMapPluginCollection
(1)   id="id"
/>
```

En el siguiente ejemplo, el archivo `companyGridCollection.xml` se utiliza para mostrar cómo utilizar el elemento backingMapPluginCollection. En este archivo, la BackingMap Customer utiliza la backingMapPluginCollection customerPlugins para configurar la BackingMap Customer con un LRUEvictor. Los elementos BackingMaps Item y OrderLine hacen referencia a labackingMapPluginCollection collection2. Cada una de estas BackingMaps tienen un conjunto LFUEvictor.

```
<?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>
```

El siguiente código de ejemplo muestra el enfoque programático para lograr la misma configuración que el archivo `companyGridCollection.xml` en el ejemplo anterior.

```
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");
```

Elemento querySchema

El elemento querySchema define las relaciones entre BackingMaps e identifica el tipo de objeto en cada correlación. Esta información la utiliza ObjectQuery para convertir series de lenguaje de consulta en llamadas de acceso a correlaciones.

- Número de apariciones: cero a ninguna
- Elemento hijo: elemento mapSchemas, elemento relationships

Elemento mapSchemas

Cada elemento querySchema tiene un elemento mapSchemas que contiene uno o más elementos mapSchema.

- Número de apariciones: una
- Elemento hijo: elemento mapSchema

Elemento mapSchema

Un elemento mapSchema define el tipo de objeto que se almacena en una BackingMap y las instrucciones para acceder a los datos.

- Número de apariciones: una o más
- Elemento hijo: ninguno

Atributos

mapName
Especifica el nombre de la BackingMap que se va a añadir al esquema. (Necesario)

valueClass
Especifica el tipo que objeto que se almacena en la parte de valor de BackingMap. (Necesario)

primaryKeyField
Especifica el nombre del atributo de clave primaria en el atributo valueClass. La clave primaria también debe almacenarse en la parte de claves de BackingMap. (Opcional)

accessType
Identifica cómo el motor de consulta realiza una introspección y accede a los datos persistentes en las instancias del objeto valueClass. Si establece el valor en `FIELD`, se realiza una introspección en los campos de clase y se añaden al esquema. Si el valor es `PROPERTY`, se utilizan los atributos asociados a los métodos get e is. El valor predeterminado es `PROPERTY`. (Opcional)

```
<mapSchema
(1)   mapName="backingMapName"
(2)   valueClass="com.mycompany.OrderBean"
(3)   primaryKeyField="orderId"
(4)   accessType="PROPERTY" | "FIELD"
/>
```

En el siguiente ejemplo, el archivo `companyGridQuerySchemaAttr.xml` se utiliza para mostrar una configuración de mapSchema de ejemplo.

```
<?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>
```

El siguiente código de ejemplo muestra el enfoque programático para lograr la misma configuración que el archivo `companyGridQuerySchemaAttr.xml` en el ejemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Definir el esquema
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);
```

Elemento relationships

Cada elemento querySchema tiene cero o un elemento relationships que contiene uno o más elementos relationship.

- Número de apariciones: cero o ninguna
- Elemento hijo: elemento relationship

Elemento relationship

Un elemento relationship define la relación entre dos BackingMaps y los atributos del atributo valueClass que enlaza la relación.

- Número de apariciones: una o más
- Elemento hijo: ninguno

Atributos

source
Especifica el nombre de valueClass del origen de una relación. (Necesario)

target
Especifica el nombre de valueClass del destino de una relación. (Necesario)

relationField
Especifica el nombre del atributo del origen valueClass que hace referencia al destino. (Necesario)

invRelationField

Especifica el nombre del atributo del destino valueClass que hace referencia al origen. Si no se especifica este atributo, la relación es de una dirección. (Opcional)

```
<mapSchema
(1)   source="com.mycompany.OrderBean"
(2)   target="com.mycompany.CustomerBean"
(3)   relationField="customer"
(4)   invRelationField="orders"
/>
```

En el siguiente ejemplo, el archivo `companyGridQuerySchemaWithRelationshipAttr.xml` se utiliza para mostrar una configuración de mapSchema de ejemplo que incluye una relación bidireccional.

```
<?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>
```

El siguiente código de ejemplo muestra el enfoque programático para lograr la misma configuración que el archivo `companyGridQuerySchemaWithRelationshipAttr.xml` en el ejemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Definir el esquema
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);
```

Elemento streamQuerySet

El elemento streamQuerySet es el elemento de nivel superior para definir un conjunto de consultas de secuencias.

- Número de apariciones: cero a muchas

- Elemento hijo: elemento stream, elemento view

Elemento stream

El elemento stream representa una secuencia para el motor de consulta de secuencias. Cada atributo del elemento stream corresponde a un método en la interfaz StreamMetadata.

- Número de apariciones: una a muchas
- Elemento hijo: elemento basic

Atributos

name
Especifica el nombre de la secuencia. Si no se especifica este atributo, la validación fallará. (Necesario)

valueClass
Especifica el tipo de clase del valor que está almacenado en la ObjectMap de la secuencia. El tipo de clase se utiliza para convertir el objeto en los sucesos de secuencia y para generar una sentencia SQL si la sentencia no se proporciona. (Necesario)

sql
Especifica la sentencia SQL de la secuencia. Si esta propiedad no se proporciona, se genera un SQL de secuencia reflejando los atributos o los métodos de descriptor de acceso del atributo valueClass o utilizando los atributos de tuple de los metadatos de entidad. (Opcional)

access
Especifica el tipo para acceder a los atributos de la clase de valor. Si establece el valor en `FIELD`, los atributos se recuperan directamente de los campos utilizando el reflejo de Java. De lo contrario, se utilizarán métodos de descriptor de acceso para leer los atributos. El valor predeterminado es `PROPERTY`. (Opcional)

```
<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"
/>
```

Elemento view

El elemento view representa una vista de consulta de secuencias. Cada elemento stream corresponde a un método de la interfaz ViewMetadata.

- Número de apariciones: una a muchas
- Elemento hijo: elemento basic, elemento id

Atributos

name
Especifica el nombre de la vista. Si no se especifica este atributo, la validación fallará. (Necesario)

sql
Especifica el SQL de la secuencia, que define la transformación de la vista. Si no se especifica este atributo, la validación fallará. (Necesario)

valueClass
Especifica el tipo de clase del valor que está almacenado en esta vista de ObjectMap. El tipo de clase se utiliza para convertir sucesos de vista en el formato de tuple correcto que sea compatible con este tipo de clase. Si no se proporciona el tipo de clase, se utiliza un formato predeterminado que sigue a la definiciones de columna en SPTSQL (Stream Processing Technology Structured Query Language). Si se definen metadatos de entidad para esta correlación de vistas, este atributo no debe utilizarse. En su lugar se debe utilizar el metadatos de entidad. (Opcional)

access
Especifica el tipo para acceder a los atributos de la clase de valor. Si establece el tipo de acceso a `FIELD`, los valores de la columna se establecen directamente en los campos utilizando el reflejo Java. De lo contrario, se utilizarán métodos de descriptor de acceso para establecer los atributos. El valor predeterminado es `PROPERTY`. (Opcional)

```
<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"
/>
```

Elemento basic

El elemento basic se utiliza para definir una correlación del nombre de atributo en la clase de valor o metadatos de entidad con la columna que se ha definido en SPTSQL.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

```
<basic
(1)   name="attributeName"
(2)   column="columnName"
/>
```

Elemento id

El elemento id se utiliza para una correlación de atributos clave.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

```
<id
(1)   name="idName"
(2)   column="columnName"
/>
```

En el siguiente ejemplo, el archivo `StreamQueryApp2.xml` se utiliza para mostrar cómo configurar los atributos de un streamQuerySet. El conjunto de consultas de secuencias _stockQuoteSQS_ tiene una secuencia y una vista. Tanto la secuencia como la vista definen su nombre, clase de valor, sql y tipo de acceso respectivamente. La secuencia también define un elemento básico, que especifica que el atributo del volumen en la clase StockQuote se correlaciona con el volumen de la transacción de la columna SQL que se ha definido en la sentencia SQL.

```
<?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>
```

Archivo `objectGrid.xsd`

Utilice el esquema XML de descriptor de ObjectGrid para configurar WebSphere eXtreme Scale.

Consulte "Archivo XML de descriptor ObjectGrid" en la página 157 si desea ver descripciones de los elementos y atributos definidos en el archivo `objectGrid.xsd`. Si desea más información sobre el archivo Spring `objectgrid.xsd`, consulte "Archivo XML de descriptor Spring" en la página 216.

```
<?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="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>
```

Archivo XML de descriptor de metadatos de entidad

El archivo de descriptor de metadatos de entidad es un archivo XML que se utiliza para definir un esquema de entidad para WebSphere eXtreme Scale. Defina todos los metadatos de entidad en el archivo XML, o defina los metadatos de entidad como anotaciones en el archivo de la clase Java de entidad. El uso primario es para las entidades que no pueden utilizar las anotaciones Java.

Utilice la configuración XML para crear metadatos de entidad que se basen en el archivo XML. Cuando se utiliza junto con la anotación, algunos de los atributos que se definen en la configuración XML alteran temporalmente las correspondientes anotaciones. Si puede alterar temporalmente un elemento, la alteración temporal es explícita en las siguientes secciones. Consulte "Archivo `emd.xsd`" en la página 189 si desea ver un ejemplo del archivo XML de descriptor de metadatos de entidad.

Elemento id

El elemento id implica que el atributo es una clave. Como mínimo se debe especificar un elemento id. Puede especificar varias claves id para utilizarlas como una clave compuesta.

Atributos

name
Especifica el nombre del atributo. El atributo debe existir en el archivo Java.

alias
Especifica el elemento alias. El valor de alias se sustituye si se utiliza junto con una entidad anotada.

Elemento basic

El elemento basic supone que el atributo es un tipo primitivo o derivadores para tipos primitivos:

- 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 Versión 5 enum

No es necesario especificar ningún atributo como basic. Los atributos del elemento basic se configuran automáticamente utilizando el reflejo.

Elemento id-class

El elemento id_class especifica una clase de clave compuesta que ayuda a encontrar entidades con claves compuestas.

Atributos

class-name
Especifica el nombre de clase, que es una id-class, que se debe utilizar con el elemento id-class.

transient

El elemento transient implica que se ignora y no se procesa. También puede sustituirse si se utiliza junto con entidades anotadas.

Atributos

name
Especifica el nombre del atributo, que se ignora.

version

El elemento transient implica que se ignora y no se procesa. También puede sustituirse si se utiliza junto con entidades anotadas.

Atributos

name
Especifica el nombre del atributo, que se ignora.

Elemento property

Utilice el elemento property para añadir propiedades a plug-ins. El nombre de la propiedad debe corresponder a un método set de la clase a la que hace referencia el bean que lo contiene.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

name
Especifica el nombre de la propiedad. El valor que se asigna a este atributo debe corresponder a un método set de la clase que se proporciona como el atributo className de bean que lo contiene. Por ejemplo, si establece el atributo className del bean en `com.ibm.MyPlugin` y el nombre de la propiedad que se proporciona es `size`, la clase `com.ibm.MyPlugin` debe tener un método setSize. (Necesario)

tipo
Especifica el tipo de la propiedad. El tipo se pasa al método set identificado por el atributo name. Los valores válidos son los primitivos Java, sus correspondientes java.lang y java.lang.String. Los atributos de nombre y tipo deben corresponder a una signatura de método del atributo className del bean. Por ejemplo, si establece el nombre como `size` y el tipo como `int`, debe haber un método setSize(int) en la clase que se especifica como el atributo className para el bean. (Necesario)

value
Especifica el valor de la propiedad. Este valor se convierte en el tipo que se especifica mediante el atributo de tipo y se utiliza como un parámetro en la llamada al método set que se identifica mediante los atributos de nombre y tipo. El valor de este atributo no se valida de ningún modo. (Necesario)

description
Describe la propiedad. (Opcional)

```
<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"
/>
```

En el siguiente ejemplo, el archivo `companyGridProperty.xml` se utiliza para mostrar cómo añadir un elemento property a un bean. En este ejemplo, un elemento property con el nombre maxSize y el tipo int se añade a un desalojador. El desalojador com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor tiene una firma de método que coincide con el método setMaxSize(int). El valor entero 499 se pasa al método setMaxSize(int) en la clase com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor.

```
<?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="Tamaño máximo del desalojador LRU"/>
   </bean>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

El siguiente código de ejemplo muestra el enfoque programático para lograr la misma configuración que el archivo `companyGridProperty.xml` en el ejemplo anterior.

```
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();
// si en su lugar se utiliza el archivo XML,
// la propiedad añadida provocará que se realice la siguiente llamada
lruEvictor.setMaxSize(449);
customerMap.setEvictor(lruEvictor);
```

Elemento backingMapPluginsCollections

El elemento backingMapPluginsCollections es un contenedor para todos los elementos backingMapPluginCollection. En el archivo `companyGridProperty.xml` de

la sección anterior, el elemento backingMapPluginCollections contiene un elemento backingMapPluginCollection con el ID `customerPlugins`.

- Número de apariciones: cero a ninguna
- Elemento hijo: elemento backingMapPluginCollection

Elemento backingMapPluginCollection

El elemento backingMapPluginCollection define los plug-ins de BackingMap y se identifica mediante el atributo **id**. Especifique el atributo pluginCollectionRef para hacer referencia a los plug-ins. Al configurar varios plug-ins BackingMaps de forma parecida, cada BackingMap puede hacer referencia al mismo elemento backingMapPluginCollection.

- Número de apariciones: cero a muchas
- Elemento hijo: elemento bean

Atributos

id Identifica la backingMapPluginCollection, a la que hace referencia el atributo pluginCollectionRef del elemento backingMap. Cada ID debe ser exclusivo. Si el valor de un atributo pluginCollectionRef no coincide con el ID de un elemento backingMapPluginCollection, la validación XML falla. Cualquier número de elementos backingMap pueden hacer referencia a un solo elemento backingMapPluginCollection. (Necesario)

```
<backingMapPluginCollection
(1)   id="id"
/>
```

En el siguiente ejemplo, el archivo `companyGridCollection.xml` se utiliza para mostrar cómo utilizar el elemento backingMapPluginCollection. En este archivo, la BackingMap Customer utiliza la backingMapPluginCollection customerPlugins para configurar la BackingMap Customer con un LRUEvictor. Los elementos BackingMaps Item y OrderLine hacen referencia a labackingMapPluginCollection collection2. Cada una de estas BackingMaps tienen un conjunto LFUEvictor.

```
<?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>
```

El siguiente código de ejemplo muestra el enfoque programático para lograr la misma configuración que el archivo `companyGridCollection.xml` en el ejemplo anterior.

```
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");
```

Elemento querySchema

El elemento querySchema define las relaciones entre BackingMaps e identifica el tipo de objeto en cada correlación. Esta información la utiliza ObjectQuery para convertir series de lenguaje de consulta en llamadas de acceso a correlaciones. Si desea más información, consulte los detalles sobre cómo definir un esquema ObjectQuery en *Guía de programación*.

- Número de apariciones: cero a ninguna
- Elemento hijo: elemento mapSchemas, elemento relationships

Elemento mapSchemas

Cada elemento querySchema tiene un elemento mapSchemas que contiene uno o más elementos mapSchema.

- Número de apariciones: una
- Elemento hijo: elemento mapSchema

Elemento mapSchema

Un elemento mapSchema define el tipo de objeto que se almacena en una BackingMap y las instrucciones para acceder a los datos.

- Número de apariciones: una o más
- Elemento hijo: ninguno

Atributos

mapName
Especifica el nombre de la BackingMap que se va a añadir al esquema. (Necesario)

valueClass
Especifica el tipo que objeto que se almacena en la parte de valor de BackingMap. (Necesario)

primaryKeyField
Especifica el nombre del atributo de clave primaria en el atributo valueClass. La clave primaria también debe almacenarse en la parte de claves de BackingMap. (Opcional)

accessType
Identifica cómo el motor de consulta realiza una introspección y accede a los

datos persistentes en las instancias del objeto valueClass. Si establece el valor en FIELD, se realiza una introspección en los campos de clase y se añaden al esquema. Si el valor es PROPERTY, se utilizan los atributos asociados a los métodos get e is. El valor predeterminado es PROPERTY. (Opcional)

```
<mapSchema
(1)   mapName="backingMapName"
(2)   valueClass="com.mycompany.OrderBean"
(3)   primaryKeyField="orderId"
(4)   accessType="PROPERTY" | "FIELD"
/>
```

En el siguiente ejemplo, el archivo companyGridQuerySchemaAttr.xml se utiliza para mostrar una configuración de mapSchema de ejemplo.

```
<?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>
```

El siguiente código de ejemplo muestra el enfoque programático para lograr la misma configuración que el archivo companyGridQuerySchemaAttr.xml en el ejemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Definir el esquema
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);
```

Elemento relationships

Cada elemento querySchema tiene cero o un elemento relationships que contiene uno o más elementos relationship.

- Número de apariciones: cero o ninguna
- Elemento hijo: elemento relationship

Elemento relationship

Un elemento relationship define la relación entre dos BackingMaps y los atributos del atributo valueClass que enlaza la relación.

- Número de apariciones: una o más
- Elemento hijo: ninguno

Atributos

source
Especifica el nombre de valueClass del origen de una relación. (Necesario)

target
Especifica el nombre de valueClass del destino de una relación. (Necesario)

relationField
Especifica el nombre del atributo del origen valueClass que hace referencia al destino. (Necesario)

invRelationField
Especifica el nombre del atributo del destino valueClass que hace referencia al origen. Si no se especifica este atributo, la relación es de una dirección. (Opcional)

```
<mapSchema
(1)   source="com.mycompany.OrderBean"
(2)   target="com.mycompany.CustomerBean"
(3)   relationField="customer"
(4)   invRelationField="orders"
/>
```

En el siguiente ejemplo, el archivo `companyGridQuerySchemaWithRelationshipAttr.xml` se utiliza para mostrar una configuración de mapSchema de ejemplo que incluye una relación bidireccional.

```
<?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>
```

El siguiente código de ejemplo muestra el enfoque programático para lograr la misma configuración que el archivo `companyGridQuerySchemaWithRelationshipAttr.xml` en el ejemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Definir el esquema
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);
```

Elemento timeBasedDBUpdate

Un elemento timeBasedDBUpdate define una configuración para un actualizador de base de datos basado en el tiempo. Un elemento timeBasedDBUpdate contiene información sobre la frecuencia con la que se recuperan los registros actualizados e insertados recientemente de la base de datos utilizando JPA (Java Persistence API), y sobre cómo actualizar los datos en las correlaciones ObjectGrid correspondientes.

- Número de apariciones: cero o ninguna
- Elemento hijo: ninguno

Atributos

entityClass
Especifica el nombre de clase de entidad utilizado para interactuar con el proveedor de JPA. El nombre de clase de entidad se utiliza para recuperar entidades JPA utilizando consultas de entidad. (Necesario)

persistenceUnitName
Especifica el nombre de unidad de persistencia de JPA para crear una fábrica del gestor de entidades de JPA. El valor predeterminado es el nombre de la primera unidad de persistencia definida en el archivo `persistence.xml`. (Opcional)

mode
Especifica la modalidad de actualización de base de datos basada en el tiempo. De forma predeterminada, la modalidad de actualización de base de datos basada en el tiempo se establece en INVALIDATE_ONLY. Un tipo INVALIDATE_ONLY indica que se deben anular las entradas en la correlación de ObjectGrid si han cambiado los correspondientes registros de la base de datos. Un tipo UPDATE_ONLY indica que se deben actualizar las entradas existentes en la correlación ObjectGrid con los valores más recientes de la base de datos. Sin embargo, todos los registros recién insertados en la base de datos se ignoran. Un tipo INSERT_UPDATE indica que se deben actualizar las entradas existentes en la correlación ObjectGrid con los valores más recientes de la base de datos. Además, todos los registros recién insertados en la base de datos se insertan en la correlación de ObjectGrid. (Opcional)

timestampField
Especifica el nombre el campo de indicación de fecha y hora. Un valor de campo de indicación de fecha y hora se utiliza para identificar la hora o la secuencia cuando se realizó la última actualización del registro del programa de fondo de base de datos. (Opcional)

jpaPropertyFactory
Identifica el nombre de clase de implementación JPAPropertyFactory o el bean de spring. La interfaz com.ibm.websphere.objectgrid.jpa.JPAPropertyFactory se utiliza para conectar una correlación de propiedad de persistencia para alterar temporalmente las propiedades de JPA predeterminadas. Utilice los beans de la infraestructura de spring es necesario establecer atributos adicionales en la instancia JPAPropertyFactory. Si desea más información, consulte “Integración con la infraestructura Spring” en la página 210.(Opcional)

```
<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"
/>
```

Elemento streamQuerySet

El elemento streamQuerySet es el elemento de nivel superior para definir un conjunto de consultas de secuencias.

- Número de apariciones: cero a muchas
- Elemento hijo: elemento stream, elemento view

Elemento stream

El elemento stream representa una secuencia para el motor de consulta de secuencias. Cada atributo del elemento stream corresponde a un método en la interfaz StreamMetadata.

- Número de apariciones: una a muchas
- Elemento hijo: elemento basic

Atributos

name
Especifica el nombre de la secuencia. Si no se especifica este atributo, la validación fallará. (Necesario)

valueClass
Especifica el tipo de clase del valor que está almacenado en la ObjectMap de la secuencia. El tipo de clase se utiliza para convertir el objeto en los sucesos de secuencia y para generar una sentencia SQL si la sentencia no se proporciona. (Necesario)

sql
Especifica la sentencia SQL de la secuencia. Si esta propiedad no se proporciona, se genera un SQL de secuencia reflejando los atributos o los métodos de descriptor de acceso del atributo valueClass o utilizando los atributos de tuple de los metadatos de entidad. (Opcional)

access
Especifica el tipo para acceder a los atributos de la clase de valor. Si establece el valor en `FIELD`, los atributos se recuperan directamente de los campos utilizando el reflejo de Java. De lo contrario, se utilizarán métodos de descriptor de acceso para leer los atributos. El valor predeterminado es `PROPERTY`. (Opcional)

```
<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"
/>
```

Elemento view

El elemento view representa una vista de consulta de secuencias. Cada elemento stream corresponde a un método de la interfaz ViewMetadata.

- Número de apariciones: una a muchas
- Elemento hijo: elemento basic, elemento id

Atributos

name
Especifica el nombre de la vista. Si no se especifica este atributo, la validación fallará. (Necesario)

sql
Especifica el SQL de la secuencia, que define la transformación de la vista. Si no se especifica este atributo, la validación fallará. (Necesario)

valueClass
Especifica el tipo de clase del valor que está almacenado en esta vista de ObjectMap. El tipo de clase se utiliza para convertir sucesos de vista en el formato de tuple correcto que sea compatible con este tipo de clase. Si no se proporciona el tipo de clase, se utiliza un formato predeterminado que sigue a la definiciones de columna en SPTSQL (Stream Processing Technology Structured Query Language). Si se definen metadatos de entidad para esta correlación de vistas, este atributo no debe utilizarse. En su lugar se debe utilizar el metadatos de entidad. (Opcional)

access
Especifica el tipo para acceder a los atributos de la clase de valor. Si establece el tipo de acceso a `FIELD`, los valores de la columna se establecen directamente en los campos utilizando el reflejo Java. De lo contrario, se utilizarán métodos de descriptor de acceso para establecer los atributos. El valor predeterminado es `PROPERTY`. (Opcional)

```
<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"
/>
```

Elemento basic

El elemento basic se utiliza para definir una correlación del nombre de atributo en la clase de valor o metadatos de entidad con la columna que se ha definido en SPTSQL.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

```
<basic
(1)   name="attributeName"
(2)   column="columnName"
/>
```

Elemento id

El elemento id se utiliza para una correlación de atributos clave.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

```
<id
(1)   name="idName"
(2)   column="columnName"
/>
```

En el siguiente ejemplo, el archivo `StreamQueryApp2.xml` se utiliza para mostrar cómo configurar los atributos de un streamQuerySet. El conjunto de consultas de secuencias _stockQuoteSQS_ tiene una secuencia y una vista. Tanto la secuencia como la vista definen su nombre, su clase de valor, sql y tipo de acceso, respectivamente. La secuencia también define un elemento basic, que especifica que el atributo volume en la clase StockQuote se correlacione con la columna SQL transactionvolume que se ha definido en la sentencia SQL.

```
<?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>
```

Archivo `emd.xsd`

Utilice la definición de esquema XML de metadatos de entidad para crear un archivo XML de descriptor y definir un esquema de entidad para WebSphere eXtreme Scale.

Consulte “Archivo XML de descriptor de metadatos de entidad” en la página 179 si desea ver la descripciones de cada elemento y atributo del archivo `emd.xsd`.

Archivo `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="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="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: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>
```

Archivo XML de descriptor de seguridad

Utilice un archivo XML de descriptor de seguridad de ObjectGrid para configurar una topología de despliegue de eXtreme Scale con la seguridad habilitada. Este tema proporciona archivos XML de ejemplo para ilustrar distintas configuraciones.

Cada elemento y atributo del archivo XML del clúster se describe en la siguiente lista. Utilice los ejemplos para aprender cómo utilizar estos elementos y atributo para configurar el entorno.

Elemento securityConfig

El elemento securityConfig es el elemento de nivel superior del archivo XML de seguridad de ObjectGrid. Este elemento configura el espacio de nombres del archivo y la ubicación del esquema. El esquema se define en el archivo `objectGridSecurity.xsd`.

- Número de apariciones: una
- Elementos hijo: security

Elemento security

Utilice el elemento security para definir una seguridad de ObjectGrid.

- Número de apariciones: una
- Elementos hijo: authenticator, adminAuthorization y systemCredentialGenerator

Atributos

securityEnabled
Habilita la seguridad para la cuadrícula cuando se establece en true. El valor predeterminado es false. Si el valor está establecido en false, la seguridad de nivel de cuadrícula está inhabilitada. Para obtener más información, consulte "Seguridad de la cuadrícula" en la página 310. (Opcional)

singleSignOnEnabled
Permite a un cliente conectarse a cualquier servidor una vez que se ha autenticado con uno de los servidores, si el valor está establecido en true. De lo contrario, un cliente debe autenticarse con cada servidor antes de que se pueda conectar el cliente. El valor predeterminado es false. (Opcional)

loginSessionExpirationTime
Especifica la cantidad de tiempo en segundos antes de que caduque el inicio de sesión. Si la sección de inicio de sesión caduca, el cliente debe volver a autenticarse. (Opcional)

adminAuthorizationEnabled
Habilita la autorización administrativa. Si el valor está establecido en true, todas las tareas administrativas necesitan autorización. El mecanismo de autorización que se utiliza se especifica mediante el valor del atributo adminAuthorizationMechanism. El valor predeterminado es false. (Opcional)

adminAuthorizationMechanism
Indica qué mecanismo de autorización utilizar. WebSphere eXtreme Scale soporta dos mecanismos de autorización: la autorización JAAS (Java Authentication and Authorization Service) y la autorización personalizada. El mecanismo de autorización JAAS utiliza el enfoque basado en la política JAAS estándar. Para especificar JAAS como mecanismo de autorización, establezca el valor en AUTHORIZATION_MECHANISM_JAAS. El mecanismo de autorización personalizada utiliza una implementación de AdminAuthorization de plug-in de usuario. Para especificar un mecanismo de autorización personalizada, establezca el valor en AUTHORIZATION_MECHANISM_CUSTOM. Para obtener más información sobre cómo se utilizan estos dos mecanismos, consulte "Autorización de cliente de aplicaciones" en la página 314. (Opcional)

El siguiente archivo `security.xml` es una configuración de ejemplo para habilitar la seguridad de la cuadrícula de eXtreme Scale.

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>
```

Elemento authenticator

Autentica los clientes en los servidores eXtreme Scale de la cuadrícula. La clase que se especifica mediante el atributo className debe implementar la interfaz com.ibm.websphere.objectgrid.security.plugins.Authenticator. El autenticador puede utilizar las propiedades para llamar a métodos en la clase que se especifica mediante el atributo className. Consulte el elemento property para obtener más información sobre la utilización de propiedades.

En el archivo de ejemplo `security.xml` anterior, la clase com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator se especifica como el autenticador. Esta clase implementa la interfaz com.ibm.websphere.objectgrid.security.plugins.Authenticator.

- Número de apariciones: cero o ninguna
- Elemento hijo: property

Atributos

className
Especifica una clase que implementa la interfaz com.ibm.websphere.objectgrid.security.plugins.Authenticator. Utilice esta clase para autenticar los clientes en los servidores de la cuadrícula de eXtreme Scale. (Necesario)

Elemento adminAuthorization

Utilice el elemento adminAuthorization para configurar el acceso administrativo a la cuadrícula.

- Número de apariciones: cero o ninguna
- Elemento hijo: property

Atributos

className
Especifica una clase que implementa la interfaz com.ibm.websphere.objectgrid.security.plugins.AdminAuthorization. (Necesario)

Elemento systemCredentialGenerator

Utilice un elemento systemCredentialGenerator para configurar un generador de credenciales del sistema. Este elemento sólo se aplica a un entorno dinámico. En el modelo de configuración dinámica, el servidor de contenedor dinámico se conecta al servidor de catálogo como un cliente de eXtreme Scale y el servidor de catálogo se puede conectar al servidor de contenedor de eXtreme Scale también como un cliente. Este generador de credenciales del sistema se utiliza para representar una fábrica de la credencial del sistema.

- Número de apariciones: cero o ninguna
- Elemento hijo: property

Atributos

className
Especifica una clase que implementa la interfaz com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. (Necesario)

Consulte el archivo `security.xml` anterior para ver un ejemplo sobre cómo utilizar systemCredentialGenerator. En este ejemplo, el generador de credenciales del sistema es un com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenCredentialGenerator, que recupera el objeto RunAs Subject de la hebra.

Elemento property

Llama a los métodos set en las clases authenticator y adminAuthorization. El nombre de la propiedad corresponde al método set del atributo className del elemento authenticator o adminAuthorization.

- Número de apariciones: cero o más
- Elemento hijo: property

Atributos

name
Especifica el nombre de la propiedad. El valor que se asigna a este atributo debe corresponder a un método set de la clase que se proporciona como el atributo className de bean que lo contiene. Por ejemplo, si el atributo className del bean se establece en com.ibm.MyPlugin, y el nombre de la propiedad suministrada es size, la clase com.ibm.MyPlugin debe tener un método setSize. (Necesario)

type
Especifica el tipo de la propiedad. El tipo del parámetro se pasa al método set identificado por el atributo name. Los valores válidos son los primitivos Java, sus correspondientes java.lang y java.lang.String. Los atributos de nombre y tipo deben corresponder a una signatura de método del atributo className del bean. Por ejemplo, si el nombre es size y el tipo es int, entonces debe existir un método setSize(int) en la clase que se especifica como el atributo className para el bean. (Necesario)

value
Especifica el valor de la propiedad. Este valor se convierte en el tipo que se especifica mediante el atributo de tipo y se utiliza como un parámetro en la llamada al método set que se identifica mediante los atributos de nombre y tipo. El valor de este atributo no se valida de ningún modo. El implementador del plug-in debe verificar que el valor que se ha pasado es válido. (Necesario)

description
Proporciona una descripción de la propiedad. (Opcional)

Si desea más información, consulte "Archivo `objectGridSecurity.xsd`".

Archivo `objectGridSecurity.xsd`

Utilice el siguiente esquema XML de seguridad de ObjectGrid para habilitar la seguridad en un despliegue de eXtreme Scale.

Consulte "Archivo XML de descriptor de seguridad" en la página 192 para ver las descripciones de los elementos y atributos definidos en el archivo `objectGridSecurity.xsd`.

```
<?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>
```

Referencia del archivo de propiedades

Los archivos de propiedades del servidor contienen valores para ejecutar los servidores de catálogo y los servidores de contenedor. Puede especificar un archivo de propiedades de servidor para una configuración autónoma o de WebSphere Application Server. Los archivos de propiedades de cliente contienen valores para el cliente.

Archivos de propiedades de ejemplo

Puede utilizar los siguientes archivos de propiedades de ejemplo que están en el directorio *extremescale_root*\properties para crear el archivo de propiedades:

- sampleServer.properties
- sampleClient.properties

Propiedades del sistema en desuso

-Dcom.ibm.websphere.objectgrid.CatalogServerProperties
La propiedad estaba en desuso en WebSphere eXtreme Scale versión 7.0. Utilice la propiedad **-Dobjectgrid.server.props**.

-Dcom.ibm.websphere.objectgrid.ClientProperties
La propiedad estaba en desuso en WebSphere eXtreme Scale versión 7.0. Utilice la propiedad **-Dobjectgrid.client.props**.

-Dobjectgrid.security.server.prop
La propiedad estaba en desuso en WebSphere eXtreme Scale versión 6.1.0.3. Utilice la propiedad **-Dobjectgrid.server.prop**.

-serverSecurityFile

Este argumento estaba en desuso en WebSphere eXtreme Scale versión 6.1.0.3. Esta opción se pasa en el script `startOgServer`. Utilice el argumento **-serverProps**.

Conceptos relacionados

"Transport Layer Security (TLC) y Secure Sockets Layer (SSL)" en la página 317
WebSphere eXtreme Scale 0soporta TCP/IP y TLS/SSL (Transport Layer Security/Secure Sockets Layer) para la comunicación segura entre el cliente y el servidor en los modelos de despliegue dinámico.

Tareas relacionadas

"Inicio de servidores de WebSphere eXtreme Scale autónomos" en la página 228
Cuando se ejecuta una configuración de WebSphere eXtreme Scale autónoma, el entorno está formado por servidores de catálogo, servidores de contenedor y procesos de cliente de eXtreme Scale. Debe configurar e iniciar manualmente estos procesos.

"Administración de WebSphere eXtreme Scale con WebSphere Application Server" en la página 242
Puede ejecutar los procesos de servicio de catálogo y de servidor de contenedor en WebSphere Application Server. El proceso para configurar estos servidores es diferente que una configuración autónoma. El servicio de catálogo se puede iniciar automáticamente en los servidores o los gestores de despliegue de WebSphere Application Server. El proceso de contenedor se inicia cuando se despliega una aplicación eXtreme Scale en el entorno WebSphere Application Server.

Archivo de propiedades de servidor

El archivo de propiedades de servidor contiene varias propiedades que definen distintos valores para el servidor como, por ejemplo, los valores de rastreo, el inicio de sesión y la configuración de seguridad. El servicio de catálogos y los servidores de contenedor utilizan el archivo de propiedades de servidor.

Archivo de propiedades de servidor de ejemplo

Puede utilizar el archivo `sampleServer.properties` que está en el directorio *raíz_extremescale*`/properties` para crear el archivo de propiedades.

Especificación de un archivo de propiedades de servidor

Puede especificar el archivo de propiedades de servidor de una de las formas siguientes. Especificar un valor utilizando uno de los elementos posteriores en la lista altera temporalmente el valor anterior. Por ejemplo, si especifica un valor de propiedad del sistema para el archivo de propiedades de servidor, las propiedades de dicho archivo alteran temporalmente los valores del archivo `objectGridServer.properties` que está en la classpath.

1. Un archivo con un nombre bien formado en la classpath. Si coloca este archivo con un nombre bien formado en el directorio actual, el archivo no se encuentra, a menos que el directorio actual esté en la classpath. El nombre que se utiliza del modo siguiente:

   ```
   objectGridServer.properties
   ```

2. Como propiedad del sistema en una configuración autónoma o de WebSphere Application Server que especifica un archivo en el directorio actual del sistema. El archivo no puede estar en la classpath:

   ```
   -Dobjectgrid.server.props=nombre_archivo
   ```

3. Como parámetro cuando se ejecuta el mandato startOgServer. Puede alterar temporalmente estas propiedades de forma manual para especificar un archivo en el directorio actual del sistema:

   `-serverProps` *`nombre_archivo`*

4. Como una alteración temporal a través de programa utilizando los métodos ServerFactory.getServerProperties y ServerFactory.getCatalogServerProperties. Los datos del objeto se rellenan con los datos procedentes de los archivos de propiedades.

Propiedades del servidor

Propiedades generales

workingDirectory
Especifica la ubicación en la que se graba la salida del servidor de contenedor. Cuando este valor no se especifica, la salida se graba en un directorio `log` dentro del directorio actual. Esta propiedad se aplica tanto al servidor de contenedor, como al servicio catálogos.

Valor predeterminado: sin valor

traceSpec
Permite el rastreo y la serie de especificación del rastreo para el servidor de contenedor. El rastreo está inhabilitado de forma predeterminada. Esta propiedad se aplica tanto al servidor de contenedor, como al servicio catálogos.

Valor predeterminado: `*=all=disabled`

traceFile
Especifica un nombre de archivo para grabar la información de rastreo. Esta propiedad se aplica tanto al servidor de contenedor, como al servicio catálogos.

systemStreamToFileEnabled
Permite al contenedor grabar la salida SystemOut, SystemErr y de rastreo en un archivo. Si esta propiedad está establecida en `false`, la salida no se graba en un archivo, y en lugar de esto, se graba en la consola.

Valor predeterminado: `true`

enableMBeans
Habilita los beans gestionados (MBean) del contenedor de ObjectGrid. Esta propiedad se aplica tanto al servidor de contenedor, como al servicio catálogos.

Valor predeterminado: `true`

serverName
Establece el nombre del servidor que se utiliza para identificar el servidor. Esta propiedad se aplica tanto al servidor de contenedor, como al servicio catálogos.

zoneName
Establezca el nombre de la zona a la que pertenece el servidor. Esta propiedad se aplica tanto al servidor de contenedor, como al servicio catálogos.

HAManagerPort
Especifica el número de puerto que utiliza el High Availability Manager. Si esta propiedad no está establecida, el servicio de catálogos genera un

puerto disponible de forma automática. Esta propiedad se aplica tanto al servidor de contenedor, como al servicio catálogos.

listenerHost
Especifica el nombre de host al que se debe enlazar el intermediario de solicitud de objetos (ORB). Esta propiedad se aplica tanto al servidor de contenedor, como al servicio catálogos.

listenerPort
Especifica el número de puerto al que se debe enlazar el intermediario de solicitud de objetos (ORB). Esta propiedad se aplica tanto al servidor de contenedor, como al servicio catálogos.

JMXServicePort
Especifica el número de puerto en el que debe estar a la escucha el MBean. Esta propiedad se aplica tanto al servidor de contenedor, como al servicio catálogos.

Propiedades de servidor de contenedor

statsSpec
Especificar la especificación stats para el servidor de contenedor.

Ejemplo:

```
all=disabled
```

memoryThresholdPercentage
Establece el umbral de memoria para el desalojo basado en memoria. El porcentaje especifica el almacenamiento dinámico máximo que se debe utilizar en máquina virtual Java (JVM) antes de que se produzca el desalojo. El valor predeterminado es `-1`, que indica que el umbral de memoria no está establecido. Si la propiedad memoryThresholdPercentage está establecida, el valor MemoryPoolMXBean se establece con el valor proporcionado. Consulte Interfaz de MemoryPoolMXBean en la especificación de la API Java si desea más información. Sin embargo, el desalojo sólo se produce, si el desalojo está habilitado en un desalojador. Para habilitar el desalojo basado en la memoria, consulte la información sobre desalojadores en *Visión general del producto*. Esta propiedad sólo se aplica a un servidor de contenedor.

catalogServiceEndPoints
Especifica los puntos finales para conectarse al clúster del servicio de catálogos. Este valor debe tener el formato `host:puerto<,host:puerto>` donde el valor de host es el valor listenerHost y el valor de puerto es el valor listenerPort del servidor de catálogo. Esta propiedad sólo se aplica a un servidor de contenedor.

Propiedades del servicio de catálogos

domainName
Especifica el nombre de dominio que se utiliza para identificar de forma exclusiva esta cuadrícula de servicio de catálogos respecto a los clientes cuando se direcciona a varios dominios. Esta propiedad se aplica al servicio de catálogos.

enableQuorum
Habilita el quórum para el servicio de catálogos. El quórum se utiliza para garantizar que una mayoría de la cuadrícula del servicio de catálogos está disponible antes de permitir la modificación en la colocación de particiones en servidores de contenedor disponibles. Para habilitar el quórum,

establezca el valor en `true` o `enabled`. El valor predeterminado es `disabled`. Esta propiedad se aplica al servicio de catálogos.

catalogClusterEndpoints
Especifica los puntos finales de la cuadrícula de servicio de catálogos para el servicio de catálogos. Esta propiedad especifica los puntos finales del servicio de catálogos para iniciar la cuadrícula del servicio de catálogos. Utilice el siguiente formato:

```
nombreservidor:nombrehost:puertocliente:puertoigual<nombreservidor:nombrehost:puertocliente:puertoigual>
```

Esta propiedad se aplica al servicio de catálogos.

heartBeatFrequencyLevel
Especifica la frecuencia con la que se producen las pulsaciones. El nivel de la frecuencia de pulsación es un equilibrio entre el uso de recursos y el tiempo de descubrimiento de anomalía. Con cuanta más frecuencia se producen las pulsaciones, más recursos se utilizan, pero las anomalías se descubren más rápidamente. Esta propiedad sólo se aplica al servicio de catálogos. Utilice uno de los valores siguientes:

- `0`: especifica un nivel de pulsación en una velocidad típica. Con este valor, la detección de la migración tras error se produce a un ritmo razonable sin un uso abusivo de recursos. (Valor predeterminado)
- `-1`: especifica un nivel de pulsación agresivo. Con este valor, las anomalías se detectan más rápidamente, pero también utiliza recursos adicionales de procesador y red. Este nivel es más propenso a perder pulsaciones cuando el servidor está ocupado.
- `1`: especifica un nivel de pulsación relajado. Con este valor, una frecuencia de pulsación reducida aumenta el tiempo para detectar las anomalías, pero también disminuye el uso de procesador y red.

Propiedades del servidor de seguridad

El archivo de propiedades de servidor también se utiliza para configurar la seguridad del servidor eXtreme Scale. Utilice un archivo de propiedades de servidor único para especificar tanto las propiedades básicas, como las propiedades de seguridad.

Propiedades de seguridad generales

securityEnabled
Habilita la seguridad del servidor de contenedor cuando se establece en `true`. El valor predeterminado es `false`. Esta propiedad debe coincidir con la propiedad securityEnabled que se especifica en el archivo `objectGridSecurity.xml` que se proporciona para el servidor de catálogo.

credentialAuthentication
Indica si este servidor soporta la autenticación de credenciales. Elija uno de los valores siguientes:

- `Nunca`: el servidor no soporta la autenticación de credenciales.
- `Soportado:`: el servidor soporta la autenticación de credenciales, si el cliente también soporta la autenticación de credenciales.
- `Necesario`: el cliente requiere la autenticación de credenciales.

Consulte “Autenticación de cliente de aplicaciones” en la página 311 si desea más detalles sobre la autenticación de credenciales.

Valores del nivel de seguridad de la capa de transporte

transportType
Especifica el tipo de transporte de servidor. Utilice uno de los valores siguientes:

- `TCP/IP`: indica que el servidor sólo soporta las conexiones TCP/IP.
- `Soportado SSL`: indica que el servidor soporta ambas conexiones, la TCP/IP y la SSL (Secure Sockets Layer). (Valor predeterminado)
- `Necesario SSL`: indica que el servidor requiere conexiones SSL.

Propiedades de configuración SSL

alias Especifica el nombre de alias del almacén de claves. Esta propiedad se utiliza si el almacén de claves tiene varios certificados de par de claves y desea seleccionar uno de los certificados.

Valor predeterminado: sin valor

contextProvider
Especifica el nombre del proveedor de contexto para el servicio de confianza. Si indica un valor que no es válido, se genera una excepción de seguridad que indica que el tipo de proveedor de contexto no es correcto.

Valores válidos: `IBMJSSE2`, `IBMJSSE`, `IBMJSSEFIPS`, etc.

protocol
Indica el tipo de protocolo de seguridad que se utiliza para el cliente. Establezca este valor de protocolo en función del proveedor de JSSE (Java Secure Socket Extension) que utilice. Si indica un valor que no es válido, se genera una excepción de seguridad que indica que el valor del protocolo no es correcto.

Valores válidos: `SSL`, `SSLv2`, `SSLv3`, `TLS`, `TLSv1`, etc.

keyStoreType
Indica el tipo de almacén de claves. Si indica un valor que no es válido, se genera una excepción de seguridad de tiempo de ejecución.

Valores válidos: `JKS`, `JCEK`, `PKCS12`, etc.

trustStoreType
Indica el tipo de almacén de confianza. Si indica un valor que no es válido, se genera una excepción de seguridad de tiempo de ejecución.

Valores válidos: `JKS`, `JCEK`, `PKCS12`, etc.

keyStore
Especifica una vía de acceso plenamente cualificada al archivo de almacén de claves.

Ejemplo:

```
etc/test/security/client.private
```

trustStore
Especifica una vía de acceso plenamente cualificada al archivo de almacén de confianza.

Ejemplo:

```
etc/test/security/server.public
```

keyStorePassword
Especifica la contraseña de serie para el almacén de claves. Puede codificar este valor o utilizar el valor real.

trustStorePassword
Especifica una contraseña de serie para el almacén de confianza. Puede codificar este valor o utilizar el valor real.

clientAuthentication
Si la propiedad se establece en true, el cliente SSL se debe autenticar. La autenticación del cliente de SSL es distinta de la autenticación del certificado de cliente. La autenticación de certificados de cliente significa autenticar un cliente en un registro de usuarios basándose en la cadena de certificados. Esta propiedad garantiza que el servidor se conecte al cliente correcto.

Valor SecureTokenManager

El valor SecureTokenManager se utiliza para proteger la serie secreta para varias autenticaciones mutuas y para proteger la señal de inicio de sesión único."Seguridad de la cuadrícula" en la página 310

secureTokenManagerType
Especifica el tipo de valor SecureTokenManager. Puede utilizar uno de los valores siguientes:

- `none (ninguno)`: indica que no se utiliza ningún gestor de señales.
- `default (predeterminado)`: indica que se utiliza el gestor de señales que se proporciona con el producto WebSphere eXtreme Scale. Debe proporcionar una configuración de almacén de claves SecureToken.
- `custom`: indica que tiene su propio gestor de señales que ha especificado con la clase de implementación SecureTokenManager.

customTokenManagerClass
Especifica el nombre de la clase de implementación SecureTokenManager, si ha especificado el valor de propiedad SecureTokenManagerType como `custom`. La clase de implementación debe tener un constructor predeterminado para el que se debe crear instancias.

customSecureTokenManagerProps
Especifica las propiedades de la clase de implementación SecureTokenManager personalizada. Esta propiedad se utiliza sólo si el valor secureTokenManagerType es `custom`. El valor se establece en el objeto SecureTokenManager con el método setProperties(String).

Configuración del almacén de claves de la señal segura

secureTokenKeyStore
Especifica el nombre de vía de acceso de archivo para el almacén de claves que almacena el par de claves pública-privada y la clave secreta.

secureTokenKeyStoreType
Especifica el tipo de almacén de claves, por ejemplo, JCKES. Puede establecer este valor basándose en el proveedor JSSE (Java Secure Socket Extension) que utilice. No obstante, este almacén de claves debe admitir claves secretas.

secureTokenKeyPairAlias
Especifica el alias del par de claves pública-privada que se utiliza para la firma y la verificación.

secureTokenKeyPairPassword
Especifica la contraseña para proteger el alias del par de claves que se utiliza para la firma y la verificación.

secureTokenSecretKeyAlias
Especifica el alias de clave secreta que se utiliza para el cifrado.

secureTokenSecretKeyPassword
Especifica la contraseña para proteger la clave secreta.

secureTokenCipherAlgorithm
Especifica el algoritmo que se utiliza para proporcionar un cifrado. Puede establecer este valor basándose en el proveedor JSSE (Java Secure Socket Extension) que utilice.

secureTokenSignAlgorithm
Especifica el algoritmo que se utiliza para firmar el objeto. Puede establecer este valor en función del proveedor JSSE que utilice.

Serie de autenticación

authenticationSecret
Especifica la serie secreta para desafiar al servidor. Cuando se inicia un servidor, debe estar presente esta serie al servidor presidente o al servidor de catálogo. Si la serie secreta coincide con lo que aparece en el servidor presidente, este servidor está autorizado para unirse.

Conceptos relacionados

"Transport Layer Security (TLC) y Secure Sockets Layer (SSL)" en la página 317
WebSphere eXtreme Scale 0soporta TCP/IP y TLS/SSL (Transport Layer Security/Secure Sockets Layer) para la comunicación segura entre el cliente y el servidor en los modelos de despliegue dinámico.

Tareas relacionadas

"Inicio de servidores de WebSphere eXtreme Scale autónomos" en la página 228
Cuando se ejecuta una configuración de WebSphere eXtreme Scale autónoma, el entorno está formado por servidores de catálogo, servidores de contenedor y procesos de cliente de eXtreme Scale. Debe configurar e iniciar manualmente estos procesos.

"Administración de WebSphere eXtreme Scale con WebSphere Application Server" en la página 242
Puede ejecutar los procesos de servicio de catálogo y de servidor de contenedor en WebSphere Application Server. El proceso para configurar estos servidores es diferente que una configuración autónoma. El servicio de catálogo se puede iniciar automáticamente en los servidores o los gestores de despliegue de WebSphere Application Server. El proceso de contenedor se inicia cuando se despliega una aplicación eXtreme Scale en el entorno WebSphere Application Server.

Archivo de propiedades de cliente

Puede crear un archivo de propiedades basándose en los requisitos para los procesos de cliente de eXtreme Scale.

Archivo de propiedades de cliente de ejemplo

Puede utilizar el archivo `sampleClient.properties` que está en el directorio *raíz_extremescale*`\properties` para crear su archivo de propiedades.

Especificación de un archivo de propiedades de cliente

Puede especificar el archivo de propiedades de cliente de una de las formas siguientes. Especificar un valor utilizando uno de los elementos posteriores en la lista altera temporalmente el valor anterior. Por ejemplo, si especifica un valor de propiedad del sistema para el archivo de propiedades de cliente, las propiedades

de dicho archivo alteran temporalmente los valores del archivo `objectGridClient.properties` que está en la classpath.

1. Un archivo con un nombre bien formado en algún lugar de la classpath. No está soportado colocar este archivo en el directorio actual del sistema.

   ```
   objectGridClient.properties
   ```

2. Como una propiedad del sistema en una configuración autónoma o en una configuración de WebSphere Application Server. Este valor puede especificar un archivo en el directorio actual del sistema, pero no un archivo en la classpath:

   ```
   -Dobjectgrid.client.props=nombre_archivo
   ```

3. Como una alteración programática utilizando el método ClientClusterContext.getClientProperties. Los datos del objeto se rellenan con los datos procedentes de los archivos de propiedades. No puede configurar las propiedades de seguridad con este método.

Propiedades de cliente

preferLocalProcess
Especifica si el proceso local es el preferido para el direccionamiento. Si está establecido en true, las solicitudes se direccionan a los fragmentos que se colocan en el mismo proceso que el cliente, cuando es preciso.

Valor predeterminado: `true`

preferLocalHost
Especifica si el host local es el preferido para el direccionamiento. Si está establecido en true, las solicitudes se direccionan a los fragmentos que se colocan en el mismo host que el cliente, cuando es preciso.

Valor predeterminado: `true`

preferZones
Especifica una lista de zonas de direccionamiento preferidas. Cada zona especificada se separa a través de una coma con el formato: `preferZones=ZoneA,ZoneB,ZoneC`

Valor predeterminado: sin valor

requestRetryTimeout
Especifica cuánto tiempo debe pasar para reintentar una solicitud (en milisegundos). Utilice uno de los siguientes valores válidos:

- Un valor de `0` indica que la solicitud debe fallar rápidamente e ignorar la lógica interna de reintentos.
- Un valor de `-1` indica que el tiempo de espera de reintento de solicitud no está establecido, lo que significa que la duración de la solicitud está regida por el tiempo de espera de la transacción. (Valor predeterminado)
- Un valor superior a `0` indica el valor de tiempo de espera de reintento de solicitud en milisegundos. Las excepciones que no se pueden generar, aunque se vuelvan a intentar como una excepción DuplicateException, se devuelven de forma inmediata. El tiempo de espera de la transacción se sigue utilizando como el tiempo de espera máximo.

Propiedades del cliente de seguridad

Propiedades de seguridad generales

securityEnabled
Habilita la seguridad de cliente de WebSphere eXtreme Scale. Este valor habilitado de seguridad debe coincidir con el valor securityEnabled en el

archivo de propiedades del servidor WebSphere eXtreme Scale. Si los valores no coinciden, se genera una excepción.

Valor predeterminado: `false`

Propiedades de configuración de la autenticación de credenciales

credentialAuthentication
Especifica el soporte de autenticación de la credencial del cliente. Utilice uno de los siguientes valores válidos:

- `Nunca`: el cliente no soporta la autenticación de credenciales.
- `Soportado`: el cliente soporta la autenticación de credenciales, si el servidor también soporta la autenticación de credenciales. (Valor predeterminado)
- `Necesario`: el cliente requiere la autenticación de credenciales.

authenticationRetryCount
Especifica el número de veces que se ha reintentado dicha autenticación, si la credencial ha caducado. Si el valor está establecido en 0, no se reintentan los intentos de autenticación.

Valor predeterminado: 3

credentialGeneratorClass
Especifica el nombre de la clase que implementa la interfaz com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Esta clase se utiliza para obtener credenciales para los clientes.

Valor predeterminado: sin valor

credentialGeneratorProps
Especifica las propiedades para la implementación de CredentialGenerator. Las propiedades se establecen en el objeto con el método setProperties(String). El valor credentialGeneratorprops sólo se utiliza si el valor de la propiedad credentialGeneratorClass no es nula.

Propiedades de configuración de la capa de transporte

transportType
Especifica el tipo de transporte del cliente. Los valores posibles son:

- `TCP/IP`: indica que el cliente sólo soporta las conexiones TCP/IP.
- `Soportado SSL`: indica que el cliente soporta tanto las conexiones TCP/IP, como las conexiones SSL (Secure Sockets Layer). (Valor predeterminado)
- `Necesario SSL`: indica que el cliente requiere las conexiones SSL.

Propiedades de configuración de SSL

alias Especifica el nombre de alias del almacén de claves. Esta propiedad se utiliza si el almacén de claves tiene varios certificados de par de claves y desea seleccionar uno de los certificados.

Valor predeterminado: sin valor

contextProvider
Especifica el nombre del proveedor de contexto para el servicio de confianza. Si indica un valor que no es válido, se genera una excepción de seguridad que indica que el tipo de proveedor de contexto no es correcto.

Valores válidos: `IBMJSSE2`, `IBMJSSE`, `IBMJSSEFIPS`, etc.

protocol
Indica el tipo de protocolo de seguridad que se utiliza para el cliente.

Establezca este valor de protocolo en función del proveedor de JSSE (Java Secure Socket Extension) que utilice. Si indica un valor que no es válido, se genera una excepción de seguridad que indica que el valor del protocolo no es correcto.

Valores válidos: `SSL`, `SSLv2`, `SSLv3`, `TLS`, `TLSv1`, etc.

keyStoreType
Indica el tipo de almacén de claves. Si indica un valor que no es válido, se produce una excepción de seguridad de tiempo de ejecución.

Valores válidos: `JKS`, `JCEK`, `PKCS12`, etc.

trustStoreType
Indica el tipo de almacén de confianza. Si indica un valor que no es válido, se genera una excepción de seguridad de tiempo de ejecución.

Valores válidos: `JKS`, `JCEK`, `PKCS12`, etc.

keyStore
Especifica una vía de acceso plenamente cualificada al archivo de almacén de claves.

Ejemplo:

```
etc/test/security/client.private
```

trustStore
Especifica una vía de acceso plenamente cualificada al archivo de almacén de confianza.

Ejemplo:

```
etc/test/security/server.public
```

keyStorePassword
Especifica la contraseña de serie para el almacén de claves. Puede codificar este valor o utilizar el valor real.

trustStorePassword
Especifica una contraseña de serie para el almacén de confianza. Puede codificar este valor o utilizar el valor real.

Conceptos relacionados

"Transport Layer Security (TLC) y Secure Sockets Layer (SSL)" en la página 317
WebSphere eXtreme Scale 0soporta TCP/IP y TLS/SSL (Transport Layer Security/Secure Sockets Layer) para la comunicación segura entre el cliente y el servidor en los modelos de despliegue dinámico.

Tareas relacionadas

"Inicio de servidores de WebSphere eXtreme Scale autónomos" en la página 228
Cuando se ejecuta una configuración de WebSphere eXtreme Scale autónoma, el entorno está formado por servidores de catálogo, servidores de contenedor y procesos de cliente de eXtreme Scale. Debe configurar e iniciar manualmente estos procesos.

"Administración de WebSphere eXtreme Scale con WebSphere Application Server" en la página 242
Puede ejecutar los procesos de servicio de catálogo y de servidor de contenedor en WebSphere Application Server. El proceso para configurar estos servidores es diferente que una configuración autónoma. El servicio de catálogo se puede iniciar automáticamente en los servidores o los gestores de despliegue de WebSphere Application Server. El proceso de contenedor se inicia cuando se despliega una aplicación eXtreme Scale en el entorno WebSphere Application Server.

Archivo de propiedades ORB

El archivo `orb.properties` se utiliza para pasar las propiedades utilizadas por el intermediario de solicitud de objetos (ORB) para modificar el comportamiento de transporte de la cuadrícula.

Ubicación

El archivo `orb.properties` se encuentra en el directorio `java/jre/lib`. Al modificar el archivo en un directorio WebSphere Application Server `java/jre/lib`, los servidores de aplicaciones configurados bajo dicha instalación también utilizan los valores del archivo.

Valores básicos

Los siguientes valores son una buena base, pero no necesariamente los mejores valores para todos los entornos. Debe comprender los valores para ayudarle a realizar una buena decisión sobre qué valores son apropiados en el entorno.

```
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=0
com.ibm.CORBA.iiop.NoLocalCopies=true
com.ibm.CORBA.NoLocalInterceptors=true
```

Descripciones de propiedad

Valores de tiempo de espera

Los siguientes valores están relacionados con la cantidad de tiempo que espera el ORB antes de abandonar una solicitud de operaciones.

Tiempo de espera de solicitud

Nombre de propiedad: com.ibm.CORBA.RequestTimeout

Valor: valor entero para el número de segundos.

Descripción: indica cuántos segundos deberá esperar una petición una respuesta antes de abandonarla. Esta propiedad influye en la cantidad de tiempo que tarda el cliente en fallar si se produce una caída de la red. Si establece este propiedad en un valor demasiado bajo, las solicitudes podrían exceder el tiempo de espera sin querer. Considere atentamente el valor de esta propiedad para impedir tiempos de espera excedidos involuntarios.

Tiempo de espera de conexión

Nombre de propiedad: com.ibm.CORBA.ConnectTimeout

Valor: valor entero para el número de segundos.

Descripción: indica cuántos segundos debe esperar un intento de conexión de socket antes de abandonar. Esta propiedad, al igual que el tiempo de espera de solicitud, puede influir en el tiempo que tarda un cliente en fallar si se produce una caída de la red. En general, establezca esta propiedad en un valor menor que el valor del tiempo de espera de solicitud porque la cantidad de tiempo para establecer unas conexiones debe ser relativamente constante.

Tiempo de espera de fragmento

Nombre de propiedad: com.ibm.CORBA.FragmentTimeout

Valor: valor entero para el número de segundos.

Descripción: indica cuántos segundos deberá esperar una solicitud de fragmento antes de abandonar. Esta propiedad es similar a la propiedad de tiempo de espera de solicitud.

Valores de agrupación de hebras

Estas propiedades limitan el tamaño de la agrupación de hebras a un número específico de hebras. Las hebras son utilizadas por el ORB para derivar las solicitudes de servidor después de que se reciban en el socket. Establecer estos valores de propiedad en valores demasiado bajos genera una profundidad de cola de socket aumentada y, posiblemente, tiempos de espera excedidos.

Multiplicidad de conexión

Nombre de propiedad: com.ibm.CORBA.ConnectionMultiplicity

Valor: valor entero para el número de conexiones entre el cliente y el servidor. El valor predeterminado es 1. Establecer un valor mayor establece la multiplexación entre varias conexiones.**Descripción:** permite al ORB utilizar varias conexiones con un servidor cualquiera. En teoría, establecer este valor debe promover paralelismos sobre las conexiones. En la práctica, el rendimiento no saca partido de la definición de la multiplicidad de conexiones. No establezca este parámetro.

Conexiones abiertas

Nombres de propiedad: com.ibm.CORBA.MinOpenConnections, com.ibm.CORBA.MaxOpenConnections

Valor: un valor entero para el número de conexiones.**Descripción:** especifica un número máximo y mínimo de conexiones abiertas. El ORB mantiene una memoria caché de conexiones que se han establecido con clientes. Estas conexiones se depuran cuando se pasa el valor com.ibm.CORBA.MaxOpenConnections. La depuración de conexiones podría provocar un pobre rendimiento en la cuadrícula.

Con posibilidad de crecimiento

Nombre de propiedad: com.ibm.CORBA.ThreadPool.IsGrowable

Valor: booleano; establecido en `true` o `false`.**Descripción:** si está habilitado, permite a la agrupación de hebras que utiliza el ORB para las solicitudes de entrada crecer más allá de los que soporta la agrupación. Si el tamaño de la agrupación se excede, se crean nuevas hebras para manejar la solicitud, pero las hebras no se agrupan.

Profundidad de cola de socket de servidor

Nombre de propiedad: com.ibm.CORBA.ServerSocketQueueDepth

Valor: un valor entero para el número de conexiones.**Descripción:** especifica la longitud de la cola de las conexiones de entrada de clientes. El ORB pone en cola las conexiones de entrada de clientes. Si la cola está llena, se rechazan las conexiones. Rechazar conexiones podría provocar un bajo rendimiento en la cuadrícula.

Tamaño de fragmento

Nombre de propiedad: com.ibm.CORBA.FragmentSize

Valor: un número entero que especifica el número de bytes. El valor predeterminado es 1024.**Descripción:** especifica el tamaño máximo de paquete que utiliza el ORB al enviar una solicitud. Si una solicitud es mayor que el límite de tamaño de fragmento, dicha solicitud se divide en fragmentos de solicitud que se envían de forma separada y se vuelven a ensamblar en el servidor. Fragmentar las solicitudes es útil en las redes no fiables donde es posible que los paquetes se tengan que volver a enviar. Sin embargo, si la red es fiable, la división de las solicitudes en fragmentos podría generar una sobrecarga.

Sin copias locales

Nombre de propiedad: com.ibm.CORBA.iiop.NoLocalCopies

Valor: booleano; establecido en `true` o `false`. **Descripción:** especifica si se pasa el ORB por referencia. El ORB utiliza la invocación pasar por valor de forma predeterminada. La invocación pasar por valor provoca basura y costes de serialización adicionales en la vía de acceso, cuando se invoca una interfaz de forma local. Mediante la definición de este valor en true, el ORB utiliza un método ″pasar por referencia″ que es más eficaz que la invocación ″pasar por valor″.

Sin interceptores locales

Nombre de propiedad: com.ibm.CORBA.NoLocalInterceptors

Valor: booleano; establecido en `true` o `false`. **Descripción:** especifica si el ORB invoca los interceptores de solicitud, incluso cuando se realizan solicitudes locales (entre procesos). Los interceptores que utiliza WebSphere eXtreme Scale son para el manejo de seguridad y rutas, que no son

necesarias si la solicitud se maneja en el proceso en el que se ejecuta. Los interceptores que se mueven entre procesos sólo son necesarios para las operaciones de llamada de procedimiento remoto (RPC). Si no se establece ningún interceptor local, puede evitar la sobrecarga adicional que implica el uso de interceptores locales.

Cuando desee aplicar la seguridad de transporte entre los clientes y servidores de ObjectGrid, deberá añadir más propiedades al archivo `orb.properties`. Si desea más información sobre estas propiedades, consulte la sección sobre el archivo `orb.properties` para el soporte de seguridad de transporte en "Transport Layer Security (TLC) y Secure Sockets Layer (SSL)" en la página 317.

Conceptos relacionados

"Transport Layer Security (TLC) y Secure Sockets Layer (SSL)" en la página 317
WebSphere eXtreme Scale 0soporta TCP/IP y TLS/SSL (Transport Layer Security/Secure Sockets Layer) para la comunicación segura entre el cliente y el servidor en los modelos de despliegue dinámico.

Tareas relacionadas

"Configuración de un intermediario de solicitud de objetos personalizado" en la página 55
Puede utilizar una versión personalizada del intermediario de solicitud de objetos (ORB) con WebSphere eXtreme Scale cuando ejecute procesos Java Platform, Standard Edition autónomos en el entorno.

"Utilización de el intermediario para solicitudes de objetos con los procesos WebSphere eXtreme Scale" en la página 30
Puede utilizar WebSphere eXtreme Scale con las aplicaciones que utilizan el intermediario de solicitud de objetos (ORB) directamente en los entornos que no contienen WebSphere Application Server o WebSphere Application Server Network Deployment.

Integración con la infraestructura Spring

Spring es una infraestructura popular para desarrollar las aplicaciones Java. WebSphere eXtreme Scale proporciona soporte para permitir a Spring gestionar las transacciones de eXtreme Scale y configurar los clientes y servidores que conforman la cuadrícula de datos en memoria desplegada.

Transacciones nativas gestionadas de Spring

Spring proporciona transacciones gestionadas por contenedor que son similares al servidor de aplicaciones Java Platform, Enterprise Edition. Sin embargo, el mecanismo Spring se puede conectar a distintas implementaciones. WebSphere eXtreme Scale proporciona una integración del gestor de transacciones que permite a Spring gestionar los ciclos de vida de transacción de ObjectGrid. Consulte la información sobre las transacciones nativas en *Guía de programación*, para buscar detalles.

Beans de ampliación gestionados de Spring y soporte de espacio de nombres

Además, eXtreme Scale se integra con Spring para habilitar a los beans de estilo Spring definidos para los puntos o plug-ins de ampliación. Esta característica proporciona configuraciones más sofisticadas y más flexibilidad para configurar los puntos de ampliación.

Además de los beans de amplicación gestionados de Spring, eXtreme Scale proporciona un espacio de nombres Spring denominado "objectgrid". Los beans y las implementaciones incorporadas están definidos previamente en este espacio de nombres, que hace que sea más fácil para los usuarios configurar eXtreme Scale. Consulte "Beans de ampliación de Spring y soporte de espacio de nombres", si desea más detalles sobre estos temas y un ejemplo sobre cómo iniciar un servidor de contenedor de eXtreme Scale utilizando las configuraciones de Spring.

Soporte de ámbito de fragmento

Con la configuración de Spring de estilo tradicional, un bean ObjectGrid puede ser un tipo singleton o un tipo de prototipo. Además, ObjectGrid soporta un nuevo ámbito denominado el ámbito de "fragmento". Si un bean está definido como ámbito de fragmento, sólo se crea un bean por fragmento. Todas las solicitudes para los beans con un ID o ids que coinciden con dicha definición de bean en el mismo fragmento generarán que una instancia de bean específica sea devuelta por el contenedor Spring.

El siguiente ejemplo muestra que un bean com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl está definido con el ámbito establecido en fragmento. Por lo tanto, sólo se crea una instancia de la clase JPAPropFactoryImpl por fragmento.

```
<bean id="jpaPropFactory" class="com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl" scope="shard" />
```

Flujo web de Spring

El flujo web de Spring almacena su estado de sesión en la sesión HTTP de forma predeterminada. Si una aplicación web se configura para utilizar eXtreme Scale para la gestión de sesiones, Spring lo utiliza automáticamente para almacenar su estado y se convierte en tolerante a errores del mismo modo que la sesión.

Empaquetado

Las extensiones Spring de eXtreme Scale están en el archivo `ogspring.jar`. Este archivo Java (JAR) debe estar en la classpath para trabajar con el soporte de Spring. Si una aplicación JEE se está ejecutando en un WebSphere Extended Deployment aumentado WebSphere Application Server Network Deployment, la aplicación deberá colocar el archivo `spring.jar` y sus archivos asociados en los módulos archivadores empresariales (EAR). También debe colocar el archivo `ogspring.jar` en la misma ubicación.

Beans de ampliación de Spring y soporte de espacio de nombres

WebSphere eXtreme Scale proporciona una característica para declarar objetos POJO (Plain Old Java Object) para utilizarlos como puntos de ampliación en el archivo `objectgrid.xml` y un método para denominar los beans y, a continuación, especificar el nombre de la clase. Normalmente, se crean las instancias de la clase especificada y estos objetos se utilizan como los plug-ins. Ahora, eXtreme Scale puede delegar en Spring para obtener las instancias de estos objetos de plug-in. Si una aplicación utiliza Spring en general será necesario que los POJO se conecten al resto de la aplicación.

En algunos casos, debe utilizar Spring para configurar determinados objetos de plug-in. Tome la siguiente configuración como ejemplo:

```
<objectGrid name="Grid">
    <bean id="TransactionCallback" className="com.ibm.websphere.objectgrid.jpa.JPATxCallback">
        <property name="persistenceUnitName" type="java.lang.String"  value="employeePU" />
    </bean>
    ...
</objectGrid>
```

La implementación de TransactionCallback incorporada, la clase com.ibm.websphere.objectgrid.jpa.JPATxCallback, se configura como la clase TransactionCallback. Esta clase se configura con una propiedad persistenceUnitName tal como se muestra en el ejemplo anterior. La clase JPATxCallback también tiene el atributo JPAPropertyFactory, que es del tipo java.lang.Object. La configuración XML de ObjectGrid no puede soportar este tipo de configuración.

La integración de Spring eXtreme Scale resuelve este problema delegando la creación de bean en la infraestructura Spring. La configuración revisada es la siguiente:

```
<objectGrid name="Grid">
    <bean id="TransactionCallback" className="{spring}jpaTxCallback"/>
    ...
</objectGrid>
```

El archivo spring para el objeto ″Grid″ contiene la siguiente información:

```
<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>
```

Aquí, TransactionCallback se especifica como `{spring}jpaTxCallback`, y los beans jpaTxCallback y jpaPropFactory se configuran en el archivo spring tal como se indica en el ejemplo anterior. La configuración de Spring hace posible configurar un bean JPAPropertyFactory como un parámetro del objeto JPATxCallback.

Fábrica de beans spring predeterminada

Cuando eXtreme Scale encuentra un plug-in o un bean de ampliación (como ObjectTransformer, Loader, TransactionCallback, etc.) con un valor de classname que empieza con el prefijo `{spring}`, eXtreme Scale utiliza el resto del nombre como un nombre de bean Spring y obtenga la instancia del bean mediante la fábrica de beans de Spring.

De forma predeterminada, si no se registró ninguna fábrica de beans para un ObjectGrid determinado, intenta encontrar un archivo `ObjectGridName_spring.xml`. Por ejemplo, si la cuadrícula se llama ″Grid″, el archivo XML se denomina `/Grid_spring.xml`. Este archivo debe estar en la classpath o en un directorio `META-INF` que está en la classpath. Si no se encuentra este archivo, eXtreme Scale construye un ApplicationContext utilizando dicho archivo y construye beans desde esa fábrica de beans.

Fábrica de beans spring personalizada

WebSphere eXtreme Scale también proporciona una API ObjectGridSpringFactory para registrar una instancia de fábrica de beans Spring para utilizar para un ObjectGrid con un nombre específico. Esta A^PI registra una instancia de BeanFactory con eXtreme Scale utilizando el siguiente método estático:

```
void registerSpringBeanFactoryAdapter(String objectGridName, Object
springBeanFactory)
```

Soporte de espacio de nombres

Desde la versión 2.0, Spring tiene un mecanismo para las ampliaciones basadas en esquema del formato XML de Spring básico y para definir y configurar beans. ObjectGrid utiliza esta nueva características para definir y configurar beans ObjectGrid. Con la ampliación del esquema XML de Spring, algunas de las implementaciones incorporadas de los plug-ins eXtreme Scale y algunos beans ObjectGrid están definidos previamente en el espacio de nombres ″objectgrid″. Al escribir los archivos de configuración de Spring, no tiene que especificar el nombre de clase completo de los programas incorporados. En lugar de esto, puede hacer referencia a los beans predefinidos.

Además, con los atributos de los beans definidos en el esquema XML, es menos probable que proporcione un nombre de atributo erróneo. La validación XML basada en el esquema XML puede capturar antes los errores de este tipo en el ciclo de desarrollo.

Estos beans definidos en las ampliaciones del esquema XML son:

- transactionManager
- register
- server
- catalog
- container
- JPALoader
- JPATxCallback
- JPAEntityLoader
- LRUEvictor
- LFUEvictor
- HashIndex

Estos beans están definidos en el esquema XML objectgrid.xsd. Este archivo XSD se suministra como un archivo `com/ibm/ws/objectgrid/spring/namespace/objectgrid.xsd` en el archivo `ogspring.jar`. Si desea descripciones detalladas del archivo XSD y los beans definidos en el archivo XSD, consulte la información sobre el archivo descriptor de Spring en *Guía de administración*.

Siga utilizando el ejemplo JPATxCallback de la sección anterior. En la sección anterior, se configura el bean JPATxCallback del modo siguiente:

```
<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>
```

Mediante esta característica del espacio de nombres, la configuración XML de spring se puede escribir del modo siguiente:

```
<objectgrid:JPATxCallback id="jpaTxCallback" persistenceUnitName="employeeEMPU"
 jpaPropertyFactory="jpaPropFactory" />

<bean id="jpaPropFactory" class="com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl"
 scope="shard">
</bean>
```

Tenga en cuenta aquí que en lugar de especificar la clase ″com.ibm.websphere.objectgrid.jpa.JPATxCallback″ como en el ejemplo anterior, directamente se utiliza el bean ″objectgrid:JPATxCallback″ definido previamente. Como puede ver, esta configuración es menos verbosa y más apta para la comprobación de errores.

Inicio del servidor de contenedor con beans de ampliación Spring

En este ejemplo, se muestra cómo iniciar un servidor ObjectGrid utilizando los beans de ampliación gestionados Spring de ObjectGrid y el soporte de espacio de nombres.

Archivo XML ObjectGrid

En primer lugar, defina un archivo XML ObjectGrid muy sencillo que contenga una ″Grid″ de ObjectGrid ″Grid″ y una correlación ″Test″. ObjectGrid tiene un plug-in ObjectGridEventListener llamado ″partitionListener″, y la correlación ″Test″ tiene un desalojador conectado llamado ″testLRUEvictor″. Tenga en cuenta que el plug-in ObjectGridEventListener y el plug-in Evictor se han configurado ambos utilizando Spring ya que sus nombres contienen ″{spring}″.

```
<?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>
```

Archivo XML de despliegue ObjectGrid

Ahora, cree un archivo XML de despliegue de ObjectGrid sencillo del modo siguiente. Divida ObjectGrid en 5 particiones, no es necesaria ninguna réplica.

```
<?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>
```

Archivo XML Spring de ObjectGrid

Ahora se utilizarán ambas características, los beans de ampliación gestionados Spring de ObjectGrid y el soporte de espacio de nombres, para configurar los beans ObjectGrid. El archivo XML spring se llama ″Grid_spring.xml″. Tenga en cuenta que se incluyen dos esquemas en el archivo XML: spring-beans-2.0.xsd se utiliza para los beans gestionados Spring y objectgrid.xsd se utiliza para los beans predefinidos en el espacio de nombres objectgrid.

```
<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>
```

Había 6 beans definidos en este archivo XML spring:

1. *objectgrid:register*: registra la fábrica de beans predeterminada para la ″Grid″ de ObjectGrid.
2. *objectgrid:server*: define un servidor ObjectGrid con el nombre ″server″. Este servidor también proporcionará un servicio de catálogos puesto que tiene un bean objectgrid:catalog que está anidado ahí.
3. *objectgrid:catalog*: define un punto final de servicio de catálogos ObjectGrid, que se establece en ″localhost:2809″.
4. *objectgrid:container*: define un contenedor ObjectGrid con un archivo XML objectgrid especificado y un archivo XML de despliegue, tal como se indicó antes. La propiedad de servidor especifica en qué servidor está alojado este contenedor.
5. *objectgrid:LRUEvictor*: define un LRUEvictor con el número de colas LRU para utilizar establecido en 31.
6. *bean partitionListener*: define un plug-in ShardListener. Esta clase es una clase conectada por los usuarios, de forma que no puede utilizar los beans predefinidos. Además, este ámbito del bean está establecido en ″shard″, que indica que sólo hay una instancia de este ShardListener por fragmento de ObjectGrid.

Inicio del servidor

El fragmento siguiente inicia el servidor ObjectGrid, que aloja tanto el servicio de contenedor, como el servicio de catálogos. Como se puede ver, el único método que se necesita llamar para iniciar el servidor es obtener un ″container″ de la fábrica de beans. Así se simplifica la complejidad de la programación moviendo la mayoría de la lógica a la configuración de Spring.

```
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();
    }
}
```

Archivo XML de descriptor Spring

Utilice un archivo XML de descriptor Spring para configurar e integrar eXtreme Scale con Spring.

En las siguientes secciones, se define cada elemento y atributo del archivo Spring `objectgrid.xsd`. El archivo Spring `objectgrid.xsd` está en el archivo `ogspring.jar` y el espacio de nombres de objectgrid `com/ibm/ws/objectgrid/spring/namespace`. Consulte "Archivo Spring `objectgrid.xsd`" en la página 221 si desea ver un ejemplo del esquema XML de descriptor.

Elemento register

Utilice el elemento register para registrar la fábrica de beans predeterminada para la cuadrícula de ObjectGrid.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

id Especifica el nombre del directorio de bean predeterminado para un ObjectGrid determinado.

gridname
Especifica el nombre de la instancia de ObjectGrid. El valor asignado a este atributo se debe corresponder a un ObjectGrid válido configurado en el archivo descriptor ObjectGrid.

```
<register
(1) id="id register"
(2) gridname="nombre ObjectGrid"
/>
```

Elemento server

Utilice el elemento server para definir un servidor eXtreme Scale, que puede alojar un contenedor, un servicio de catálogo, o ambos.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

id Especifica el nombre del servidor eXtreme Scale.

tracespec
Indica el tipo de rastreo y habilita el rastreo y la especificación para el servidor.

tracefile
Proporciona la vía de acceso y el nombre del traceFile para crear y utilizar.

statspec
Indica la especificación de estadística para el servidor.

jmxport
Designa el número de puerto no utilizado a través del cual desde habilitar las conexiones JMX/RMI. JMX habilita la supervisión y la gestión de sistemas remotos.

isCatalog
Especifica si el servidor determinado aloja un servicio de catalogo. El valor predeterminado es `false`.

name
Especifica el nombre del servidor.

```
<server
(1) id="id servidor"
(2) tracespec="la especificación de rastreo de servidor"
(3) tracefile="el archivo de rastreo de servidor"
(4) statspec="la especificación de estadística de servidor"
(5) jmxport="número de puerto JMX"
(6) isCatalog="true"|"false"
(7) name="el nombre de servidor"
/>
```

Elemento catalog

Utilice el elemento catalog para direccionar a los servidores de contenedor de la cuadrícula de datos.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

host
Especifica el nombre de sistema principal de la estación de trabajo en la que se ejecuta el servicio de catálogo.

port
Especifica el número de puerto emparejado al nombre de sistema principal para determinar el puerto del servicio de catálogo al que se puede conectar el cliente.

```
<catalog
(1) host="nombre de host de servicio de catálogo"
(2) port="número de puerto de servicio de catálogo"
/>
```

Elemento container

Utilice el elemento container para almacenar los propios datos.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

objectgridxml
Especifica la vía de acceso y el nombre del archivo XML de descriptor para utilizar que especifica las características para el ObjectGrid, incluidos las correlaciones, la estrategia de bloqueo y los plug-ins.

deploymentxml
Especifica la vía de acceso y el nombre del archivo XML que se utiliza con el XML de descriptor para determinar el particionamiento, la réplica, el número de contenedores iniciales y otros valores.

server
Especifica el servidor en el que se aloja el contenedor.

```
<server
(1) objectgridxml="el archivo XML de descriptor de objectgrid"
(2) deploymentxml ="el archivo XML de descriptor de despliegue de objectgrid "
(3) server="la referencia del servidor "
/>
```

Elemento JPALoader

Utilice el elemento JPALoader para sincronizar la memoria caché de ObjectGrid con un almacén de datos de programa de fondo existente cuando se utiliza la API ObjectMap.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

entityClassName
Habilita el uso de JPA como, por ejemplo, EntityManager.persist y EntityManager.find. El atributo **entityClassName** es necesario para JPALoader.

preloadPartition
Especifica el número de partición en el que se inicia la precarga de correlación. Si el valor es inferior a 0, o mayor que (totalNumberOfPartition – 1), no se inicia la precarga de correlación.

```
<JPALoader
(1) entityClassName="el nombre de clase de entidad"
(2) preloadPartition ="int"
/>
```

Elemento JPATxCallback

Utilice el elemento JPATxCallback para coordinar las transacciones JPA y ObjectGrid.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

persistenceUnitName
Crea un JPA EntityManagerFactory y localiza los metadatos de entidad JPA en el archivo persistence.xml. El atributo **persistenceUnitName** es necesario.

jpaPropertyFactory
Especifica la fábrica para crear una correlación de propiedad de persistencia para alterar temporalmente las propiedades predeterminadas de persistencia. Este atributo debe hacer referencia a un bean.

exceptionMapper
Especifica el plug-in ExceptionMapper que se puede utilizar para las funciones de correlación de excepciones específicas de JPA o específicas de base de datos. Este atributo debe hacer referencia a un bean.

```
<JPATxCallback
(1) persistenceUnitName="el nombre de la unidad de persistencia JPA"
(2) jpaPropertyFactory ="referencia de bean JPAPropertyFactory"
(3) exceptionMapper="referencia de bean ExceptionMapper"
/>
```

Elemento JPAEntityLoader

Utilice el elemento JPAEntityLoader para sincronizar la memoria caché de ObjectGrid con un almacén de datos de programa de fondo existente cuando se utiliza la API EntityManager.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

entityClassName
Habilita el uso de JPA como, por ejemplo, EntityManager.persist y EntityManager.find. El atributo **entityClassName** es opcional para el elemento JPAEntityLoader. Si el elemento no se ha configurado, se utiliza la clase de entidad configurada en la correlación de entidades ObjectGrid. Se debe utilizar la misma clase para ObjectGrid EntityManager y para el proveedor JPA.

preloadPartition
Especifica el número de partición en el que se inicia la precarga de correlación. Si el valor es menor que 0, o mayor que (totalNumberOfPartition – 1) no se inicia la precarga de correlación.

```
<JPAEntityLoader
(1) entityClassName="el nombre de clase de entidad"
(2) preloadPartition ="int"
/>
```

Elemento LRUEvictor

Utilice el elemento LRUEvictor para decidir qué entradas desalojar cuando una correlación excede su número máximo de entradas.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

maxSize
Especifica el total de entradas de una cola hasta que deba intervenir el desalojador.

sleepTime
Establece el tiempo en segundos entre los barridos de un desalojador sobre las colas para determinar cualquier acción necesaria en la correlación.

numberOfLRUQueues
Especifica el valor sobre cuántas colas debe examinar el desalojador para evitar tener una sola cola que sea el tamaño de toda la correlación.

useMemoryUsageThresholdEviction
Determina los valores de desalojo basados en cuánta memoria utiliza una entrada. El valor predeterminado es `false`.

```
<LRUEvictor
(1) maxSize="int"
(2) sleepTime ="segundos"
(3) numberOfLRUQueues ="int"
(4) useMemoryUsageThresholdEviction ="true"|"false"
/>
```

Elemento LFUEvictor

Utilice el elemento LFUEvictor para determinar qué entradas desalojar cuando una correlación excede su número máximo de entradas.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

maxSize
Especificar el total de entradas que están permitidas en cada almacenamiento dinámico hasta que deba actuar el desalojador.

sleepTime
Establece el tiempo en segundos entre los barridos de un desalojador sobre los almacenamientos dinámico de correlación para determinar cualquier acción necesaria en la correlación.

numberOfHeaps
Especifica el valor sobre cuántos almacenamientos dinámico debe examinar el desalojador para evitar tener un único almacenamiento dinámico que tenga el tamaño de toda la correlación.

useMemoryUsageThresholdEviction
Determina los valores de desalojo basados en cuánta memoria utiliza una entrada.

```
<LFUEvictor
(1) maxSize="int"
(2) sleepTime ="segundos"
(3) numberOfHeaps ="int"
(4) useMemoryUsageThresholdEviction ="true"|"false"
/>
```

Elemento HashIndex

Utilice el elemento HashIndex con el reflejo Java para hacer una introspección dinámica de los objetos almacenados en una correlación cuando se actualizan los objetos.

- Número de apariciones: cero a muchas
- Elemento hijo: ninguno

Atributos

name
Especifica el nombre del índice, que debe ser exclusivo para cada correlación.

attributeName
Especifica el nombre del atributo para indexar. Para los índices de acceso del campo, el nombre del atributo es equivalente al nombre de campo. Para los índices de acceso de propiedad, el nombre de atributo es el nombre de propiedad compatible con JavaBean.

rangeIndex
Indica si está habilitada la indexación de rangos. El valor predeterminado es `false`.

fieldAccessAttribute
Se utiliza para las correlaciones sin entidad. Se utiliza el método getter para acceder a los datos. El valor predeterminado es `false`. Si especifica el valor como `true`, se accede al objeto a través de los campos directamente.

POJOKeyIndex
Se utiliza para las correlaciones sin entidad. El valor predeterminado es `false`. Si especifica el valor como `true`, el índice realiza una introspección del objeto en la parte de clave de la correlación, que es práctico cuando la clave es una clave compuesta y el valor no tiene que tener ninguna clave incorporada. Si no establece el valor o si especifica el valor como `false`, el índice realiza una introspección del objeto en la parte de valor de la correlación.

```
<HashIndex
(1) name="nombre de índice"
(2) attributeName="nombre de atributo"
(3) rangeIndex ="true"|"false"
```

```
(4) fieldAccessAttribute ="true"|"false"

(5) POJOKeyIndex ="true"|"false"
/>
```

Archivo Spring objectgrid.xsd

Utilice el archivo Spring objectgrid.xsd para integrar eXtreme Scale con Spring para gestionar las transacciones eXtreme Scale y configurar clientes y servidores.

Consulte “Archivo XML de descriptor Spring” en la página 216 si desea descripciones de los elementos y atributos definidos en el archivo Spring objectgrid.xsd.

Archivo Spring 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: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="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>
```

Utilización de WebSphere Real Time

Puede utilizar WebSphere Real Time con WebSphere eXtreme Scale. Mediante la habilitación de WebSphere Real Time, puede obtener una recogida de basura más predecible junto con un tiempo de respuesta estable y coherente y un rendimiento de transacciones en un entorno autónomo de eXtreme Scale.

¿Por qué utilizar WebSphere Real Time?

WebSphere eXtreme Scale crea muchos objetos temporales que se asocian a cada transacción. Estos objetos temporales se ocupan de peticiones, respuestas, secuencias de registro y sesiones. Sin WebSphere Real Time, el tiempo de respuesta de la transacción puede ascender hasta miles de milisegundos. Sin embargo, el uso de WebSphere Real Time con WebSphere eXtreme Scale puede aumentar la eficacia de la recogida de basura y reducir el tiempo de respuesta en un 10% del tiempo de respuesta de la configuración autónoma.

Habilitación de WebSphere Real Time

Instale WebSphere Real Time y el WebSphere eXtreme Scale autónomo en los sistemas en los que tiene previsto ejecutar eXtreme Scale. Establezca la variable de entorno JAVA_HOME para indicar un Java SE Runtime Environment (JRE) estándar.

Establezca la variable de entorno JAVA_HOME para indicar al WebSphere Real Time instalado. A continuación, habilite WebSphere Real Time del modo siguiente.

1. Edite el archivo de instalación autónomo `objectgridRoot/bin/setupCmdLine.sh | .bat` eliminando el comentario de la siguiente línea.

```
WXS_REAL_TIME_JAVA="-Xrealtime -Xgcpolicy:metronome
-Xgc:targetUtilization=80"
```

2. Guarde el archivo.

Ahora, ha habilitado WebSphere Real Time. Si desea inhabilitar WebSphere Real Time, puede volver a añadir el comentario a la misma línea.

Procedimientos recomendados

WebSphere WebSphere Real Time permite a las transacciones eXtreme Scale tener un tiempo de respuesta más predecible. Los resultados muestran que la desviación de un tiempo de respuesta de una transacción eXtreme Scale mejora significativamente con WebSphere Real Time, en comparación con el Java estándar con su recogida de basura predeterminada. La habilitación de WebSphere Real Time con eXtreme Scale es óptima si la estabilidad y el tiempo de respuesta de la aplicación son esenciales.

Los mejores procedimientos descritos en esta sección explican cómo hacer más eficaz a WebSphere eXtreme Scale a través del ajuste y de las prácticas de código, en función de la carga esperada.

- Establezca el nivel correcto de uso de procesador para la aplicación y la recogida de basura.

  WebSphere Real Time proporciona la capacidad para controlar el uso del procesador, de forma que el impacto de la recogida de basura en la aplicación está controlado y minimizado. Utilice el parámetro `-Xgc:targetUtilization=NN` para especificar el `NN` porcentaje del procesador que es utilizado por la aplicación cada 20 segundos. El valor predeterminado para WebSphere eXtreme Scale es 80%, pero puede modificar el script en el archivo `objectgridRoot/bin/setupCmdLine.sh` para definir un número distintos como, por ejemplo, `70`, que proporciona más capacidad de procesador a la recogida de basura. Despliegue los suficientes servidores para mantener la carga del procesador por debajo del 80% para las aplicaciones.

- Establezca un tamaño mayor de memoria de almacenamiento dinámico.

  WebSphere Real Time utiliza más memoria que el Java típico, así que planifique WebSphere eXtreme Scale con una memoria de almacenamiento dinámico grande y establezca el tamaño del almacenamiento dinámico cuando inicie los servidores y contenedores de catálogo con el parámetro `–jvmArgs –XmxNNNM` en el mandato ogStartServer. Por ejemplo, podría utilizar el parámetro `–jvmArgs –Xmx500M` para iniciar los servidores de catálogo y utilizar el tamaño de memoria apropiado para iniciar los contenedores. Puede establecer el tamaño de la memoria en un 60-70% del tamaño de datos esperado por JVM. Si no establece este valor, se podría generar un error OutOfMemoryError. De forma opcional, también puede utilizar el parámetro `–jvmArgs –Xgc:noSynchronousGCOnOOM` para impedir el comportamiento nondeterministic cuando la JVM agota la memoria.

- Ajuste las hebras para la recogida de basura.

  WebSphere eXtreme Scale crea muchos objetos temporales asociados a cada transacción y a hebras de llamada de procedimiento remoto (RPC). La recogida de basura tiene ventajas de rendimiento si el sistema tiene los suficientes ciclos de procesador. El número predeterminado de hebras es 1. Puede cambiar el número de hebras con el argumento `–Xgcthreads n`. El valor sugerido de este argumento es el número de núcleos que están disponibles con consideración del número de máquinas virtuales Java por sistema.

- Ajuste el rendimiento para las aplicaciones de corta ejecución con WebSphere eXtreme Scale.

WebSphere Real Time se ajusta para las aplicaciones de larga ejecución. Normalmente, debe ejecutar las transacciones continuas de WebSphere eXtreme Scale durante dos horas para obtener datos de rendimiento fiables. Puede utilizar el parámetro `–Xquickstart` para mejorar el rendimiento de las aplicaciones de corta ejecución. Este parámetro indica al compilador JIT (just-in-time) que utilice el nivel inferior de optimización.

- Minimice la cola de cliente de WebSphere eXtreme Scale y la transmisión del cliente de WebSphere eXtreme Scale.

  La principal ventaja de utilizar WebSphere eXtreme Scale con WebSphere Real Time es tener un tiempo de respuesta de transacción muy fiable, que normalmente tiene varios tiempos de mejoras de magnitud de orden en la desviación del tiempo de respuesta de transacción. Las peticiones de cliente en cola y la transmisión de solicitud de cliente a través de otro software impacta en el tiempo de respuesta que está más allá del control de WebSphere Real Time y WebSphere eXtreme Scale. Debe cambiar las hebras y los parámetros de sockets para mantener una carga fija sin problemas sin ningún retardo significativo y reducir la profundidad de la cola.

- Escriba aplicaciones WebSphere eXtreme Scale para utilizar las hebras de WebSphere Real Time.

  Sin modificar la aplicación, puede obtener un tiempo de respuesta de transacción de WebSphere eXtreme Scale muy fiable con varias mejoras de magnitud de orden en la desviación del tiempo de respuesta. Puede explotar de forma adicional la ventaja de hebras de las aplicaciones transaccionales de la hebra Java regular en RealtimeThread que proporciona un mejor control en la prioridad de la hebras y una planificación del control.

  Actualmente, la aplicación incluye el siguiente código.

  ```
  public class WXSCacheAppImpl extends Thread implements WXSCacheAppIF
  ```

  De forma opcional, puede sustituir este código por lo siguiente.

  ```
  public class WXSCacheAppImpl extends RealtimeThread implements
  WXSCacheAppIF
  ```

Capítulo 7. Administración del entorno

La administración del entorno incluye el inicio y la parada de servidores en la modalidad autónoma y también dentro de WebSphere Application Server. También puede utilizar WebSphere eXtreme Scale como un gestor de sesiones dentro de un entorno WebSphere Application Server.

Por qué y cuándo se efectúa esta tarea

Tipos de servidor

WebSphere eXtreme Scale tiene dos tipos de servidores: *servidores de catálogos* y *servidores de contenedor*. Los servidores de catálogos controlan la colocación de fragmentos y descubren y supervisan los servidores de contenedor. Varios servidores de catálogos comprimen de forma conjunta el *servicio de catálogo*. Los servidores de contenedor son las Máquinas virtuales Java que almacenan los datos de aplicación para la cuadrícula.

Modalidad autónoma

La modalidad autónoma hace referencia a una configuración de WebSphere eXtreme Scale que se ejecuta por si mismo, sin ningún otro producto de servidor de aplicaciones.

Ejecución dentro de WebSphere Application Server

Cuando se ejecuta WebSphere eXtreme Scale encima de WebSphere Application Server, los servidores de catálogos se inician automáticamente en los servidores WebSphere Application Server. Para iniciar los servidores de contenedor, debe empaquetar y desplegar la aplicación con los archivos XML de ObjectGrid.

Establecer la disponibilidad de un ObjectGrid

El estado de disponibilidad de una instancia de ObjectGrid determina qué peticiones se pueden procesar en un momento dado.

Existen cuatro estados de disponibilidad:

- EN LÍNEA
- INMOVILIZADO
- FUERA DE LÍNEA
- PRECARGA

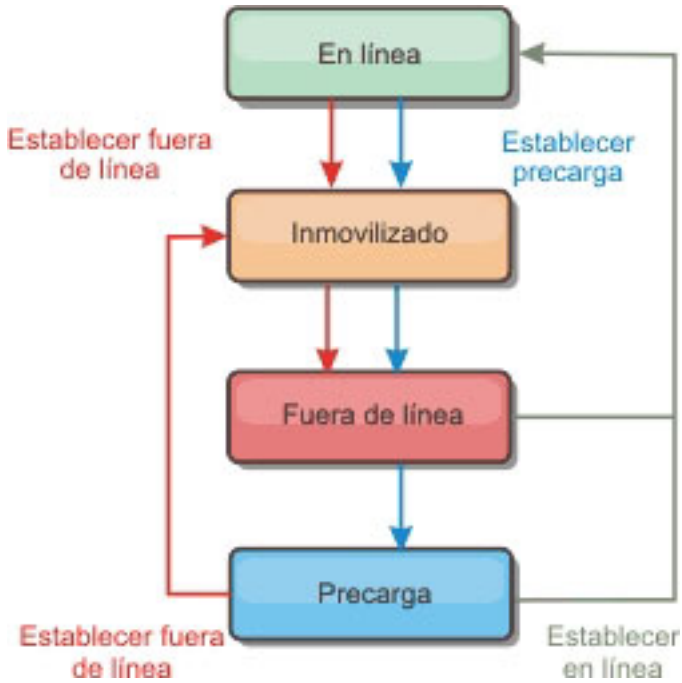


Figura 5. Estados de disponibilidad de un ObjectGrid

Establecer el estado de disponibilidad

El estado de disponibilidad predeterminado para un ObjectGrid es EN LÍNEA. Un ObjectGrid EN LÍNEA puede procesar las peticiones de un cliente típico de eXtreme Scale. Sin embargo, las solicitudes de un cliente de precarga se rechazan mientras el ObjectGrid está EN LÍNEA.

El estado INMOVILIZADO es un estado de transición. un ObjectGrid que está INMOVILIZADO estará pronto en estado FUERA DE LÍNEA. Mientras está INMOVILIZADO, un ObjectGrid puede procesar transacciones pendientes. Sin embargo se rechazarán las nuevas transacciones. un ObjectGrid puede permanecer en QUIESCE hasta 30 segundos. Una vez transcurrido este intervalo, el estado de disponibilidad pasará a FUERA DE LÍNEA.

un ObjectGrid en el estado FUERA DE LÍNEA rechazará todas las transacciones.

El estado PRECARGA se puede utilizar para cargar datos en ObjectGrid desde un cliente de precarga. Mientras ObjectGrid está en estado PRELOAD, sólo un cliente de precarga puede confirmar transacciones respecto a ObjectGrid. Las demás transacciones se rechazarán.

Utilice la interfaz StateManager para establecer el estado de disponibilidad de un ObjectGrid. Los ObjectGrids cliente, así como los ObjectGrids del lado del servidor, pueden tener su estado de disponibilidad alterado mediante el uso de la interfaz StateManager. El siguiente código demuestra cómo cambiar el estado de disponibilidad de un ObjectGrid de cliente.

```
ClientClusterContext client = ogManager.connect("localhost:2809", null, null);
ObjectGrid myObjectGrid = ogManager.getObjectGrid(client, "myObjectGrid");
StateManager stateManager = StateManagerFactory.getStateManager();
stateManager.setObjectGridState(AvailabilityState.OFFLINE, myObjectGrid);
```

Cada fragmento del ObjectGrid realiza la transición del estado deseado cuando se llama al método setObjectGridState en la interfaz StateManager. Cuando se devuelve el método, todos los fragmentos de el ObjectGrid deben estar en el estado adecuado.

Utilice un plug-in ObjectGridEventListener para cambiar el estado de disponibilidad de un ObjectGrid del lado del servidor. Cambie sólo el estado de disponibilidad de un ObjectGrid del lado del servidor, si ObjectGrid tiene una única partición. Si el ObjectGrid tiene varias particiones, se llama al método shardActivated en cada fragmento primario, que genera llamadas superfluas para cambiar el estado del ObjectGrid

```
public class OGListener implements ObjectGridEventListener, ObjectGridEventGroup.ShardEvents {
public void shardActivated(ObjectGrid grid) {
            StateManager stateManager = StateManagerFactory.getStateManager();
            stateManager.setObjectGridState(AvailabilityState.PRELOAD, grid);
    }
}
```

Puesto que INMOVILIZADO es un estado de transición, no puede utilizar la interfaz StateManager para colocar un ObjectGrid en el estado INMOVILIZADO. Un ObjectGrid pasa a través de este estado al estado FUERA DE LÍNEA.

Recuperación del estado de disponibilidad

Utilice el método getObjectGridState de la interfaz StateManager para recuperar el estado de disponibilidad de un ObjectGrid determinado.

```
StateManager stateManager = StateManagerFactory.getStateManager();
AvailabilityState state = stateManager.getObjectGridState(inventoryGrid);
```

El método getObjectGridState elige un primario aleatorio dentro de ObjectGrid y devuelve su AvailabilityState. Como todos los fragmentos de ObjectGrid deben estar en el mismo estado de disponibilidad o en transición al mismo estado de disponibilidad, este método proporciona un resultado aceptable para el estado de disponibilidad actual de ObjectGrid.

Estados de disponibilidad adecuados para diversas solicitudes

Una solicitud se rechazará si un ObjectGrid no está en el estado de disponibilidad adecuado para dar soporte a esa solicitud. Se genera una excepción AvailabilityException siempre que se rechaza una solicitud.

El atributo initialState

Puede utilizar el atributo initialState en un ObjectGrid para indicar su estado de inicio. Normalmente, cuando un ObjectGrid finaliza su inicialización, está disponible para el direccionamiento. El estado podrá cambiarse más adelante para impedir que el tráfico se direccione a un ObjectGrid. Si es necesario inicializar el ObjectGrid, pero no está disponible inmediatamente, puede utilizar el atributo initialState.

El atributo initialState se establece en el archivo XML de configuración de ObjectGrid. El estado predeterminado es EN LÍNEA. Los valores válidos son:

- EN LÍNEA (valor predeterminado)
- PRECARGA
- FUERA DE LÍNEA

Consulte la documentación de la API AvailabilityState si desea más información.

Si se ha establecido initialState en un ObjectGrid, el estado debe volver a establecerse de forma explícita en en línea o el ObjectGrid permanecerá no disponible. Generará AvailabilityExceptions.

Utilización del atributo initialState para la precarga

Si el ObjectGrid se precarga con datos, puede haber un periodo de tiempo entre el momento en que el ObjectGrid está disponible y el cambio a un estado de precarga para bloquear el tráfico del cliente. Para evitar este periodo de tiempo, el estado inicial en un ObjectGrid se puede establecer en PRECARGA. El ObjectGrid finalizará toda la inicialización necesaria, pero bloqueará el tráfico hasta que el estado cambie y permita que se produzca la precarga.

Los estados PRECARGA y FUERA DE LÍNEA bloquean el tráfico, pero debe utilizarse el estado PRECARGA si desea iniciar una precarga.

Migración tras error y equilibrio

Si una réplica se promueve a fragmento primario, no utilizará el valor initialState. Si el primario se traslada para volver a equilibrarlo, el valor de initialState no se utilizará porque los datos se copian a la nueva ubicación primara antes de que se complete el traslado. Si no se configura la réplica, el fragmento primario pasa al valor de initialState, si se produce una migración tras error y se debe colocar el nuevo primario.

Inicio de servidores de WebSphere eXtreme Scale autónomos

Cuando se ejecuta una configuración de WebSphere eXtreme Scale autónoma, el entorno está formado por servidores de catálogo, servidores de contenedor y procesos de cliente de eXtreme Scale. Debe configurar e iniciar manualmente estos procesos.

Antes de empezar

Puede iniciar los servidores WebSphere eXtreme Scale en un entorno que no tiene WebSphere Application Server instalado. Si utiliza WebSphere Application Server, consulte “Administración de WebSphere eXtreme Scale con WebSphere Application Server” en la página 242.

Qué hacer a continuación

Detenga los procesos de eXtreme Scale. Si desea más información, consulte “Detención de servidores de eXtreme Scale autónomos” en la página 239.

Conceptos relacionados

"Inicio y parada de servidores eXtreme Scale seguros" en la página 327
A menudo, los servidores necesitan ser seguros para el entorno de despliegue, que requiere una configuración específica.

Referencia relacionada

"Archivo de propiedades de servidor" en la página 197
El archivo de propiedades de servidor contiene varias propiedades que definen distintos valores para el servidor como, por ejemplo, los valores de rastreo, el inicio de sesión y la configuración de seguridad. El servicio de catálogos y los servidores de contenedor utilizan el archivo de propiedades de servidor.

"Archivo de propiedades de cliente" en la página 203
Puede crear un archivo de propiedades basándose en los requisitos para los procesos de cliente de eXtreme Scale.

"Referencia del archivo de propiedades" en la página 196
Los archivos de propiedades del servidor contienen valores para ejecutar los servidores de catálogo y los servidores de contenedor. Puede especificar un archivo de propiedades de servidor para una configuración autónoma o de WebSphere Application Server. Los archivos de propiedades de cliente contienen valores para el cliente.

"Script startOgServer" en la página 235
El script startOgServer inicia servidores. Puede utilizar diversos parámetros al iniciar los servidores para habilitar el rastreo, especificar números de puerto, etc.

"Registros y rastreo" en la página 303
Puede utilizar los registros y el rastreo para supervisar y resolver problemas del entorno. Los registros están en distintas ubicaciones en función de su configuración. Es posible que sea necesario proporcionar el rastreo para un servidor cuando se trabaja con el servicio de soporte IBM.

Inicio del servicio de catálogo en un entorno autónomo

Debe iniciar el servicio de catálogo manualmente cuando utilice un entorno distribuido de WebSphere eXtreme Scale que no ejecute WebSphere Application Server.

Antes de empezar

Si utiliza WebSphere Application Server, el servicio de catálogo se inicia automáticamente dentro de uno de los procesos existentes. Consulte Inicio del servicio de catálogo en WebSphere Application Server si desea más información.

Por qué y cuándo se efectúa esta tarea

El servicio de catálogo puede ejecutarse en un único proceso o pueden incluir varios servidores de catálogo para formar la cuadrícula del servidor de catálogo. En un entorno de producción, es necesaria una cuadrícula de servidor de catálogo para la alta disponibilidad. Si desea más información sobre cómo configurar una cuadrícula de servidor de catálogo, consulte la información sobre la agrupación en clúster del servicio de catálogo en *Visión general del producto*. El servicio de catálogo, ya esté dentro de una cuadrícula o en un proceso único, se inicia utilizando el script `startOgServer`. También puede especificar parámetros adicionales en el script para enlazar el intermediario de solicitud de objetos (ORB) con un host y puerto específicos, especificar el dominio o habilitar la seguridad.

Cuando llame al mandato start, utilice el script `startOgServer.sh` en las plataformas Unix o `startOgServer.bat` en Windows.

- **Inicio de un único proceso de servidor de catálogo**

  Para iniciar un servidor de catálogo único, escriba los siguientes mandatos desde la línea de mandatos:

  1. Vaya al directorio bin:

     ```
     cd objectgridRoot/bin
     ```

  2. Ejecute el mandato startOgServer:

     ```
     startOgServer.bat|sh catalogServer
     ```

  Para obtener una lista de todos los parámetros de línea de mandatos disponibles, consulte "Script startOgServer" en la página 235. No utilice una sola máquina virtual Java (JVM) para ejecutar el servicio de catálogo en un entorno de producción. Si el servicio de catálogo falla, ningún cliente nuevo podrá direccionarse al eXtreme Scale desplegado, y no se podrá añadir ninguna instancia nueva de ObjectGrid al dominio. Por estos motivos, deberá iniciar un conjunto de máquinas virtuales Java para ejecutar una cuadrícula de servidor de catálogo.

- **Inicio de una cuadrícula de servidor de catálogo de varios procesos**

  Para iniciar un conjunto de servidores para que ejecuten un servicio de catálogo, debe utilizar la opción **-catalogServiceEndPoints** en el script`startOgServer`. Este argumento acepta una lista de puntos finales de servicio de catálogo con el formato `serverName:hostName:clientPort:peerPort`. Cada atributo se define de la manera siguiente:

  serverName
  Especifica un nombre para identificar el proceso que está iniciando.

  hostName
  Especifica el nombre de host para el sistema donde se inicia el servidor.

  clientPort
  Especifica el puerto que se utiliza para la comunicación de la cuadrícula de catálogo de igual.

  peerPort
  Especifica el puerto que se utiliza para la comunicación de la cuadrícula de catálogo de igual.

  En el siguiente ejemplo se muestra cómo iniciar la primera de tres máquinas virtuales Java para alojar un servicio de catálogo:

  1. Vaya al directorio bin:

     ```
     cd objectgridRoot/bin
     ```

  2. Ejecute el mandato startOgServer:

     ```
     startOgServer.bat|sh cs1 -catalogServiceEndpoints
     cs1:MyServer1.company.com:6601:6602,
     cs2:MyServer2.company.com:6601:6602,
     cs3:MyServer3.company.com:6601:6602
     ```

     En este ejemplo, se inicia el servidor `cs1` en el host `MyServer1.company.com`. Este nombre de servidor es el primer argumento que se pasado al script. Durante la inicialización del usuario `cs1`, los parámetros catalogServiceEndpoints se examinan para determinar qué puertos se asignan para este proceso. La lista también se utiliza para permitir al servidor `cs1` aceptar conexiones de otros servidores: `cs2` y `cs3`.

  3. Para iniciar los servidores de catálogo restantes de la lista, pase los siguientes argumentos al script startOgServer. Inicio del servidor `cs2` en el host `MyServer2.company.com`:

```
startOgServer.bat|sh cs2 -catalogServiceEndpoints
cs1:MyServer1.company.com:6601:6602,
cs2:MyServer2.company.com:6601:6602,
cs3:MyServer3.company.com:6601:6602
```

Inicio de `cs3` en `MyServer3.company.com`:

```
startOgServer.bat|sh cs3 -catalogServiceEndpoints
cs1:MyServer1.company.com:6601:6602,
cs2:MyServer2.company.com:6601:6602,
cs3:MyServer3.company.com:6601:6602
```

Importante: Inicio de todos los servidores de catálogo en paralelo.

Debe iniciar los servidores de catálogos que están en una cuadrícula en paralelo, porque cada servidor hace una pausa para esperar que otros servidores de catálogo se unan al grupo principal. Un servidor de catálogo que se configura para una cuadrícula no se inicia hasta que identifica a otros miembros del grupo. Con el tiempo, el servidor de catálogo excede el tiempo de espera si no hay otros servidores disponibles.

- **Enlace de ORB a un host y puerto específicos**

  Aparte de los puertos definidos en el argumento**catalogServiceEndpoints**, cada servicio de catálogo también utiliza un intermediario para solicitudes de objetos (ORB) para aceptar conexiones de clientes y contenedores. De forma predeterminada, el ORB está a la escucha en el puerto `2809` del host local. Si desea enlazar el ORB con un host y puerto específicos en una JVM de servicio de catálogo, utilice los argumentos **-listenerHost** y **-listenerPort**. En el siguiente ejemplo se muestra cómo iniciar un servidor de catálogo de JVM simple con su ORB enlazado al puerto 7000 en MyServer1.company.com:

```
startOgServer.sh catalogServer -listenerHost MyServer1.company.com
-listenerPort 7000
```

Cada contenedor y cliente de eXtreme Scale deben proporcionarse con los datos de punto final de ORB de servicio de catálogo. Los clientes sólo necesitan un subconjunto de estos datos, aunque como mínimo debe utilizar dos puntos finales para la alta disponibilidad.

- **Denominación del dominio**

  Un nombre de dominio no es necesario al iniciar un servicio de catálogo. El nombre de dominio predeterminado es `defaultDomain`. Para proporciona un nombre al dominio, utilice la opción **-domain**. En el siguiente ejemplo se muestra cómo iniciar una JVM de servicio de catálogo único con el nombre de dominio `myDomain`.

```
startOgServer.sh catalogServer -domain myDomain
```

- **Iniciar un servicio de catálogo seguro**

  Puede iniciar un servicio de catálogo seguro proporcionando los siguientes argumentos:

  -clusterSecurityFile y -clusterSecurityUrl
  Estos argumentos especifican el archivo `objectGridSecurity.xml`, que describe las propiedades de seguridad que son comunes para todos los servidores (incluidos los servidores de catálogo y los servidores de contenedor). Un ejemplo de propiedad es la configuración de autenticador que representa el mecanismo de autenticación y el registro de usuarios.

  -serverProps
  Especifica el archivo de propiedades que contiene las propiedades de seguridad específicas del servidor. El nombre de archivo especificado

para esta propiedad tiene el formato de vía de acceso de archivo sencillo, por ejemplo, `c:/tmp/og/catalogserver.props`.

Si desea un ejemplo sobre cómo iniciar un servicio de catálogo seguro, consulte el paso 2 de la guía de aprendizaje de seguridad de Java SE en *Visión general del producto*.. Para obtener un ejemplo del archivo `objectGridSecurity.xml`, consulte "Archivo XML de descriptor de seguridad" en la página 192.

Conceptos relacionados

"Inicio y parada de servidores eXtreme Scale seguros" en la página 327
A menudo, los servidores necesitan ser seguros para el entorno de despliegue, que requiere una configuración específica.

Referencia relacionada

"Script startOgServer" en la página 235
El script startOgServer inicia servidores. Puede utilizar diversos parámetros al iniciar los servidores para habilitar el rastreo, especificar números de puerto, etc.

"Registros y rastreo" en la página 303
Puede utilizar los registros y el rastreo para supervisar y resolver problemas del entorno. Los registros están en distintas ubicaciones en función de su configuración. Es posible que sea necesario proporcionar el rastreo para un servidor cuando se trabaja con el servicio de soporte IBM.

Inicio de procesos de contenedor

Puede iniciar eXtreme Scale desde la línea de mandatos, utilizando una topología de despliegue, o utilizando un archivo `server.properties`.

Por qué y cuándo se efectúa esta tarea

Para iniciar un proceso de contenedor, necesita un archivo XML de ObjectGrid. El archivo XML de ObjectGrid especifica qué servidores eXtreme Scale aloja el contenedor. Asegúrese de que el contenedor esté equipado para alojar cada ObjectGrid en el XML que le pase. Todas las clases que estas ObjectGrids requieren deben estar en la vía de acceso de clases para el contenedor. Si desea más información sobre el archivo XML de ObjectGrid, consulte "Archivo `objectGrid.xsd`" en la página 174.

- **Inicie el proceso de contenedor desde la línea de mandatos.**
  1. En la línea de mandatos, vaya al directorio bin:
     ```
     cd objectgridRoot/bin
     ```
  2. Ejecute el siguiente mandato:
     ```
     startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

  Importante: En el contenedor, la opción **-catalogServiceEndPoints** se utiliza para hacer referencia al host y al puerto del intermediario de solicitud de objetos (ORB) en el servicio de catálogo. El servicio de catálogo utiliza las opciones **-listenerHost** y **-listenerPort** para especificar el host y el puerto para el enlace ORB o acepta el enlace predeterminado. Cuando inicie un contenedor, utilice la opción **-catalogServiceEndPoints** para hacer referencia a los valores pasados en las opciones **-listenerHost** y **-listenerPort** en el servicio de catálogo. Si las opciones **-listenerHost** y **-listenerPort** no se utilizan cuando se inicia el servicio de catálogo, el ORB enlaza con el puerto 2809 del host local para el servicio de catálogo. No utilice la opción **-catalogServiceEndPoints** para hacer referencia a los hosts y puertos que se pasaron en la opción **-catalogServiceEndPoints** en el servicio de catálogo. En el servicio de catálogo, la opción

-catalogServiceEndPoints se utiliza para especificar los puertos necesarios para la configuración de servidor estático.
Este proceso se identifica por c0, el primer argumento pasado al script. Utilice `companyGrid.xml` para iniciar el contenedor. Si el ORB del servidor de catálogo se ejecuta en un host distinto que el del contenedor o utiliza un puerto no predeterminado, debe utilizar el argumento **-catalogServiceEndPoints** para conectarse al ORB. Para este ejemplo, suponga que un servicio de catálogo simple se ejecuta en el puerto `2809` en `MyServer1.company.com`

- **Inicie el contenedor utilizando una política de despliegue.**

  Aunque no es necesario, se recomienda una política de despliegue durante el inicio del contenedor. La política de despliegue se utiliza para configurar el particionamiento y la réplica para eXtreme Scale. La política de despliegue también se puede utilizar para influir en el comportamiento de la colocación. Como el ejemplo anterior no ha proporcionado un archivo de política de despliegue, el ejemplo recibe todos los valores predeterminados con relación a la réplica, al réplica y a la colocación. Por ello, las correlaciones de CompanyGrid están en un mapSet. El mapSet no está particionado ni replicado. Si desea más información sobre los archivos de política de despliegue, consulte "Archivo XML de descriptor de la política de despliegue" en la página 152. En el siguiente ejemplo se utiliza el archivo `companyGridDpReplication.xml` para iniciar una JVM de contenedor, la JVM c0:

  1. En la línea de mandatos, vaya al directorio bin:

     ```
     cd objectgridRoot/bin
     ```

  2. Ejecute el siguiente mandato:

     ```
     startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml
     -deploymentPolicyFile ../xml/companyGridDpReplication.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

  Nota: Si ha almacenado clases Java en un directorio específico, en lugar de alterar el script StartOgServer, podrá lanzar el servidor con argumentos del modo siguiente: `-jvmArgs -cp C:\ . . . \DirectoryPOJOs\POJOs.jar` . En el archivo `companyGridDpReplication.xml`, un único conjunto de correlaciones contiene todas las correlaciones. Este mapSet está dividido en 10 particiones. Cada partición tiene una réplica síncrona y ninguna réplica asíncrona. Cualquier contenedor que utilice la política de despliegue de `companyGridDpReplication.xml` emparejado con el archivo XML de ObjectGrid `companyGrid.xml` también puede alojar fragmentos de CompanyGrid. Inicie otra JVM de contenedor, la JVM c1:

  1. En la línea de mandatos, vaya al directorio bin:

     ```
     cd objectgridRoot/bin
     ```

  2. Ejecute el siguiente mandato:

     ```
     startOgServer.sh c1 -objectGridFile ../xml/companyGrid.xml
     -deploymentPolicyFile ../xml/companyGridDpReplication.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

  Cada política de despliegue contiene uno o más elementos objectgridDeployment. Cuando se inicia un contenedor, éste publica su política de despliegue en el servicio de catálogo. El servicio de catálogo examina cada elemento objectgridDeployment. Si el atributo objectgridName coincide con el atributo objectgridName del elemento objectgridDeployment recibido anteriormente, se ignora el último elemento objectgridDeployment. El primer elemento objectgridDeployment recibido para un atributo objectgridName específico se utiliza como elemento maestro. Por ejemplo, suponga que la JVM c2 utiliza una política de despliegue que divide el mapSet en un número de particiones distinto:

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>
```

Ahora, puede iniciar una tercera JVM, la JVM c2:

1. En la línea de mandatos, vaya al directorio bin:

   `cd` *`objectgridRoot`*`/bin`

2. Ejecute el siguiente mandato:

```
startOgServer.sh c2 -objectGridFile ../xml/companyGrid.xml
-deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml
-catalogServiceEndPoints MyServer1.company.com:2809
```

El contenedor de la JVM c2 se inicia con una política de despliegue que especifica 5 particiones para mapSet1. Sin embargo, el servicio de catálogo ya mantiene la copia maestra de objectgridDeployment en CompanyGrid. Cuando se inició la JVM c0, se especificó que existen 10 particiones para este mapSet. Puesto que era el primera contenedor para iniciarse y publicar su política de despliegue, su política de despliegue pasa a ser la maestra. Por lo tanto, se ignora cualquier valor de atributo de objectgridDeployment que sea igual a `CompanyGrid` en una política de despliegue posterior.

- **Inicie un contenedor utilizando un archivo de propiedades de servidor.**

  Puede utilizar un archivo de propiedades de servidor para configurar el rastreo y la seguridad en un contenedor. Ejecute los siguientes mandatos para iniciar el contenedor c3 con un archivo de propiedades de servidor:

  1. En la línea de mandatos, vaya al directorio bin:

     `cd` *`objectgridRoot`*`/bin`

  2. Ejecute el siguiente mandato:

```
startOgServer.sh c3 -objectGridFile ../xml/companyGrid.xml
-deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml
-catalogServiceEndPoints MyServer1.company.com:2809
-serverProps ../serverProps/server.properties
```

A continuación, aparece un ejemplo de archivo `server.properties`:

server.properties

```
workingDirectory=
traceSpec=*=all=disabled
systemStreamToFileEnabled=true
enableMBeans=true
memoryThresholdPercentage=50
```

Éste es el archivo de propiedades de servidor básico que no tiene la seguridad habilitada. Si desea más información sobre el archivo `server.properties`, consulte “Archivo de propiedades de servidor” en la página 197.

Conceptos relacionados

"Inicio y parada de servidores eXtreme Scale seguros" en la página 327
A menudo, los servidores necesitan ser seguros para el entorno de despliegue, que requiere una configuración específica.

Referencia relacionada

"Script startOgServer"
El script startOgServer inicia servidores. Puede utilizar diversos parámetros al iniciar los servidores para habilitar el rastreo, especificar números de puerto, etc.

"Registros y rastreo" en la página 303
Puede utilizar los registros y el rastreo para supervisar y resolver problemas del entorno. Los registros están en distintas ubicaciones en función de su configuración. Es posible que sea necesario proporcionar el rastreo para un servidor cuando se trabaja con el servicio de soporte IBM.

Script startOgServer

El script startOgServer inicia servidores. Puede utilizar diversos parámetros al iniciar los servidores para habilitar el rastreo, especificar números de puerto, etc.

Finalidad

Puede utilizar el script startOgServer para iniciar servidores.

Ubicación

El script startOgServer está en el directorio bin del directorio raíz, por ejemplo:

```
cd objectgridRoot/bin
```

Nota: Si tiene clases Java almacenadas en un directorio específico, en lugar de alterar el script StartOgServer, puede lanzar el servidor con argumentos del modo siguiente: `-jvmArgs -cp C:\ . . . \DirectoryPOJOs\POJOs.jar`

.

Uso para servidores de catálogo

Para iniciar un servidor de catálogo:

Windows

```
startOgServer.bat <servidor> [options]
```

UNIX

```
startOgServer.sh <servidor>[options]
```

Para iniciar un servidor de catálogo configurado predeterminado, emplee los siguientes mandatos:

Windows

```
startOgServer.bat catalogServer
```

UNIX

```
startOgServer.sh catalogServer
```

Opciones para iniciar servidores de catálogo

Parámetros para iniciar un servidor de catálogo:

-catalogServiceEndPoints <servidor:hostServidor:puertoCliente:puertoIgual, servidor:hostServidor:puertoCliente:puertoIgual>
En el contenedor, la opción **-catalogServiceEndpoints** se utiliza para hacer referencia al host y puerto de ORB (Object Request Broker) del servicio de catálogo.

-clusterSecurityFile <archivo xml de seguridad de clúster>
Especifica la ubicación del archivo XML de descriptor de seguridad.

-clusterSecurityUrl <URL de xml de seguridad de clúster>

-domain <nombre de dominio>

-listenerHost <nombre de host>
Valor predeterminado: `localhost`

Especifica el host del escucha para la comunicación con IIOP (Internet Inter-ORB Protocol).

-listenerPort <puerto>
Valor predeterminado: `2809`

Especifica el puerto de escucha para la comunicación con IIOP.

-serverProps <archivo de propiedades de servidor>
Especifica el archivo de propiedades que contiene las propiedades de seguridad específicas del servidor. El nombre de archivo especificado para esta propiedad está en formato de vía de acceso de archivo sencillo, como `c:/tmp/og/catalogserver.props`.

-JMXServicePort <puerto>
Valor predeterminado: `1099`

Especifica el número de puerto para la comunicación con JMX (Java Management Extensions).

-traceSpec <especificación de rastreo>
Especifica una serie que especifica el ámbito del rastreo que está habilitado cuando se inicia el servidor.

Ejemplo:

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <archivo de rastreo>
Especifica la vía de acceso a un archivo en el que guardar información de rastreo.

Ejemplo: `../logs/c4Trace.log`

-timeout <segundos>
Especifica un número de segundos antes de que el inicio del servidor exceda el tiempo de espera.

-script <archivo de script>

-jvmArgs <argumento de JVM>
Especifica un conjunto de argumentos de JVM. Cada parámetro después del parámetro -**jvmArgs** se utiliza para iniciar la JVM (Java Virtual Machine) del

servidor. Cuando se utilice el parámetro **-jvmArgs**, asegúrese de que sea el último argumento del script opcional especificado.

Ejemplo: -jvmArgs -Xms256M -Xmx1G

Uso para servidores de contenedor Windows

```
startOgServer.bat <servidor> -objectgridFile <archivo xml>
-deploymentPolicyFile <archivo xml> [options]
```

Windows

```
startOgServer.bat <servidor> -objectgridUrl <URL de xml>
-deploymentPolicyUrl <URL de xml> [options]
```

UNIX

```
startOgServer.sh <servidor> -objectgridFile <archivo xml>
-deploymentPolicyFile <archivo xml> [options]
```

UNIX

```
 startOgServer.sh <servidor> -objectgridUrl <URL de xml>
-deploymentPolicyUrl <URL de xml> [options]
```

Opciones para servidores de contenedor

-catalogServiceEndPoints<nombreHost:puerto,nombreHost:puerto>
Valor predeterminado: localhost:2809

-deploymentPolicyFile <archivo xml de política de despliegue>
Especifica la vía de acceso del archivo de política de despliegue.

Ejemplo: ../xml/SimpleDP.xml

-deploymentPolicyUrl <url de xml de política de despliegue>

-listenerHost <nombre de host>
Valor predeterminado: localhost

Especifica el host del escucha para la comunicación con IIOP (Internet Inter-ORB Protocol).

-listenerPort <puerto>
Valor predeterminado: 2809

Especifica el puerto de escucha para la comunicación con IIOP.

-serverProps <archivo de propiedades de servidor>
Especifica la vía de acceso al archivo de propiedades del servidor.

Ejemplo:../security/server.props

-zone <nombre de zona>
Especifica la zona que se va a utilizar para todos los contenedores dentro del servidor.

-traceSpec <especificación de rastreo>
Especifica una serie que especifica el ámbito del rastreo que está habilitado cuando se inicia el servidor.

Ejemplo:

- ObjectGrid=all=enabled
- ObjectGrid*=all=enabled

-traceFile <archivo de rastreo>
Especifica la vía de acceso a un archivo en el que guardar información de rastreo.

Ejemplo: `../logs/c4Trace.log`

-timeout <segundos>
Especifica un número de segundos antes de que el inicio del servidor exceda el tiempo de espera.

-script <archivo de script>

-jvmArgs <argumento de JVM>
Especifica un conjunto de argumentos de JVM. Cada parámetro después del parámetro -**jvmArgs** se utiliza para iniciar la JVM (Java Virtual Machine) del servidor. Cuando se utilice el parámetro **-jvmArgs**, asegúrese de que sea el último argumento del script opcional especificado.

Ejemplo: `-jvmArgs -Xms256M -Xmx1G`

Conceptos relacionados

"Inicio y parada de servidores eXtreme Scale seguros" en la página 327
A menudo, los servidores necesitan ser seguros para el entorno de despliegue, que requiere una configuración específica.

Tareas relacionadas

"Inicio de servidores de WebSphere eXtreme Scale autónomos" en la página 228
Cuando se ejecuta una configuración de WebSphere eXtreme Scale autónoma, el entorno está formado por servidores de catálogo, servidores de contenedor y procesos de cliente de eXtreme Scale. Debe configurar e iniciar manualmente estos procesos.

"Inicio del servicio de catálogo en un entorno autónomo" en la página 229
Debe iniciar el servicio de catálogo manualmente cuando utilice un entorno distribuido de WebSphere eXtreme Scale que no ejecute WebSphere Application Server.

"Inicio de procesos de contenedor" en la página 232
Puede iniciar eXtreme Scale desde la línea de mandatos, utilizando una topología de despliegue, o utilizando un archivo `server.properties`.

Configuración de puertos TCP en modalidad autónoma

Una máquina virtual Java que aloje una instancia de servicio de catálogo requiere cuatro puertos. Dos puertos son para uso interno, el tercer puerto se utiliza para los fragmentos de clientes y contenedor para comunicarse con IIOP (Internet Inter-ORB Protocol), y el cuarto puerto se utilizar para la comunicación JMX (Java Management Extensions).

Por qué y cuándo se efectúa esta tarea

Puntos finales de máquina virtual Java (JVM) de servicio de catálogo

eXtreme Scale utiliza principalmente IIOP para comunicarse entre máquinas virtuales Java. Las máquinas virtuales Java de servicio de catálogo son las únicas máquinas virtuales Java que requieren la configuración explícita de puertos para los servicios IIOP y los puertos de servicios de grupos. Los puertos internos se especifican utilizando la opción de la línea de mandatos **-catalogServiceEndPoints**:

`-catalogServiceEndPoints <servidor:host:puerto:puerto,servidor:host:puerto:puerto>`

Con la opción de la línea de mandatos **-catalogServiceEndPoints**, puede configurar dos puertos por servidor. Los puertos IIOP están configurados utilizando las siguientes opciones de línea de mandatos:

```
-listenerHost <nombre_host>
-listenerPort <puerto>
```

Cuando se inicia cada JVM de servicio de catálogo, especifique el conjunto completo de puntos finales de servicio de catálogo junto a un único puerto de escucha para esa JVM.

Puntos finales de JVM de contenedor

Las máquinas virtuales Java de contenedor utilizan dos puertos. Un puerto es para uso interno y el otro puerto es para la comunicación IIOP. En general, las máquinas virtuales Java de contenedor encuentran puertos no utilizados y, a continuación, los configuran para utilizar dos puertos creados dinámicamente. Este proceso automático minimiza la configuración. Sin embargo, cuando se utilizan cortafuegos o si se desea configurar explícitamente los puertos, puede utilizar una opción de líneas de mandatos para especificar el puerto del intermediario de solicitud de objetos (ORB) que se debe utilizar:

```
-listenerHost <nombre_host>
-listenerPort <puerto>
```

Con estas opciones de línea de mandatos, puede especificar el nombre de host (importante para el enlace con la tarjeta de red correcta) y el puerto que se debe especificar para dicha JVM. Puede especificar el puerto interno con el argumento de la línea de mandatos **-haManagerPort**. Sin embargo, para tener la configuración más sencilla, puede dejar que el tiempo de ejecución elija los puertos.

1. Inicie el primer servidor de catálogo en el hostA. A continuación, se muestra un ejemplo del mandato:

   ```
   ./startOgServer.sh cs1 -listenerHost hostA -listenerPort 2809
   -catalogServiceEndPoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
   ```

2. Inicie el segundo servidor de catálogo en el hostB. A continuación, se muestra un ejemplo del mandato:

   ```
   ./startOgServer.sh cs2 -listenerHost hostB -listenerPort 2809
   -catalogServiceEndPoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
   ```

   Los clientes sólo necesitan saber los puntos finales del receptor del servicio de catálogo. Los clientes recuperan los puntos finales para las máquinas virtuales Java de contenedor, que están en las máquinas virtuales Java que alojan los datos, automáticamente del servicio de catálogo. Para conectarse al servicio de catálogo del ejemplo anterior, el cliente debe pasar la lista siguiente de pares `host:puerto` a la API de conexión:

   ```
   hostA:2809,hostB:2809
   ```

   Para iniciar una JVM de contenedor para utilizar el servicio de catálogos de ejemplo, utilice el siguiente mandato:

   ```
   ./startOgServer.sh c0 -catalogServiceEndPoints hostA:2809,hostB:2809
   ```

Detención de servidores de eXtreme Scale autónomos

Puede utilizar el script `stopOgServer.sh` para Unix o el script `stopOgServer.bat` para Windows para detener procesos de servidor.

- Detenga los procesos de contenedor de eXtreme Scale.
  1. En la línea de mandatos, vaya al directorio bin.

     ```
     cd objectGridRoot/bin
     ```

2. Ejecute el script stopOgServer para detener el servidor. El ejemplo siguiente detiene el servidor c0:

   ```
   stopOgServer.sh c0 -catalogServiceEndPoints MyServer1.company.com:2809
   ```

   Atención: En el contenedor, la opción **-catalogServiceEndPoints** se utiliza para hacer referencia al host y al puerto del intermediario de solicitud de objetos (ORB) en el servicio de catálogo. El servicio de catálogo utiliza las opciones **-listenerHost** y **-listenerPort** para especificar el host y el puerto para el enlace ORB o acepta el enlace predeterminado. Al detener un contenedor, utilice la opción **-catalogServiceEndPoints** para hacer referencia a los valores pasados en **-listenerHost** y **-listenerPort** en el servicio de catálogo. Si las opciones **-listenerHost** y**-listenerPort** no se utilizan al iniciar el servicio de catálogo, ORB se enlaza al puerto 2809 en el host local para el servicio de catálogo.

   No utilice la opción **-catalogServiceEndPoints** para hacer referencia a los hosts y puertos que se pasaron en la opción **-catalogServiceEndPoints** en el servicio de catálogo. En el servicio de catálogo, utilice la opción **-catalogServiceEndPoints** para especificar los puertos que son necesarios para la configuración del servidor estático.

- Detenga los procesos del servicio de catálogo de eXtreme Scale.
  1. En la línea de mandatos, vaya al directorio bin.

     ```
     cd objectGridRoot/bin
     ```

  2. Ejecute el script stopOgServer para detener el servidor.

     ```
     stopOgServer.sh catalogServer -bootstrap MyServer1.company.com:6601
     ```

  Identifique el proceso para detener mediante el argumento catalogServer, el primer argumento pasado en el script. Para este ejemplo, suponga que el servicio de catálogo se ha iniciado en el dominio MyServer1.company.com con el valor de clientAccessPort 6601.

- Habilite el rastreo para el proceso de detención de servidor. Si el contenedor no puede detenerse, puede habilitar el rastreo para ayudar a realizar la depuración del problema. Para habilitar el rastreo durante la detención de un servidor, añada los parámetros **-traceSpec** y **-traceFile** a los mandatos de detención. El parámetro **-traceSpec** especifica el tipo de rastreo que se va a habilitar y el parámetro **-traceFile** especifica la vía de acceso y el nombre del archivo que se va a crear y utilizar para los datos de rastreo.
  1. En la línea de mandatos, vaya al directorio bin.

     ```
     cd objectGridRoot/bin
     ```

  2. Ejecute el script stopOgServer con el rastreo habilitado.

     ```
     stopOgServer.sh c4 -catalogServiceEndPoints MyServer1.company.com:2809
      -traceFile ../logs/c4Trace.log -traceSpec ObjectGrid=all=enabled
     ```

     Una vez que se ha obtenido el rastreo, busque los errores relacionados con los conflictos del puerto, clases que faltan, archivos XML incorrectos o que faltan, o cualquier seguimiento de la pila. Las especificaciones de rastreo de inicio sugeridas son:
     - ObjectGrid=all=enabled
     - ObjectGrid*=all=enabled

     Para todas las opciones de especificación de rastreo, consulte “Opciones de rastreo” en la página 306.

Referencia relacionada

"Script stopOgServer"
El script stopOgServer detiene servidores.

Script stopOgServer

El script stopOgServer detiene servidores.

Finalidad

Puede utilizar el script stopOgServer para detener servidores.

Ubicación

El script stopOgServer está en el directorio bin del directorio raíz, por ejemplo:
`cd objectgridRoot/bin`

Uso

Para detener un servidor de catálogo:

Windows

```
stopOgServer.bat <servidor> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [opciones]
```

UNIX

```
stopOgServer.sh <server> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [opciones]
```

Para detener un servidor de contenedor de ObjectGrid: Windows

```
stopOgServer.bat <servidor> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [opciones]
```

UNIX

```
startOgServer.sh -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [opciones]
```

Opciones

-clientSecurityFile <archivo de propiedades de seguridad>

-traceSpec <especificación de rastreo>
Especifica una serie que especifica el ámbito del rastreo que está habilitado cuando se inicia el servidor.

Ejemplo:

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <archivo de rastreo>
Especifica la vía de acceso a un archivo en el que guardar información de rastreo.

Ejemplo: `../logs/c4Trace.log`

-jvmArgs <argumentos de JVM>
Cada parámetro después de la opción -jvmArgs se utiliza para iniciar la JVM

(Java Virtual Machine) del servidor. Cuando se utilice la opción -jvmArgs, asegúrese de que sea el último argumento del script opcional especificado.

Tareas relacionadas

"Detención de servidores de eXtreme Scale autónomos" en la página 239
Puede utilizar el script `stopOgServer.sh` para Unix o el script `stopOgServer.bat` para Windows para detener procesos de servidor.

Administración de WebSphere eXtreme Scale con WebSphere Application Server

Puede ejecutar los procesos de servicio de catálogo y de servidor de contenedor en WebSphere Application Server. El proceso para configurar estos servidores es diferente que una configuración autónoma. El servicio de catálogo se puede iniciar automáticamente en los servidores o los gestores de despliegue de WebSphere Application Server. El proceso de contenedor se inicia cuando se despliega una aplicación eXtreme Scale en el entorno WebSphere Application Server.

Referencia relacionada

"Archivo de propiedades de servidor" en la página 197
El archivo de propiedades de servidor contiene varias propiedades que definen distintos valores para el servidor como, por ejemplo, los valores de rastreo, el inicio de sesión y la configuración de seguridad. El servicio de catálogos y los servidores de contenedor utilizan el archivo de propiedades de servidor.

"Archivo de propiedades de cliente" en la página 203
Puede crear un archivo de propiedades basándose en los requisitos para los procesos de cliente de eXtreme Scale.

"Referencia del archivo de propiedades" en la página 196
Los archivos de propiedades del servidor contienen valores para ejecutar los servidores de catálogo y los servidores de contenedor. Puede especificar un archivo de propiedades de servidor para una configuración autónoma o de WebSphere Application Server. Los archivos de propiedades de cliente contienen valores para el cliente.

Inicio del proceso de servicio de catálogo en un entorno de WebSphere Application Server

Un único servicio de catálogo de eXtreme Scale puede iniciarse automáticamente en un entorno de WebSphere Application Server r WebSphere Application Server Network Deployment. Puede configurar el servicio de catálogo de modo que se ejecute en cualquier proceso dentro de la célula de WebSphere. Un único servicio de catálogo que no está en clúster es aceptable para entornos de desarrollo. Para un entorno de producción, deberá utilizar una cuadrícula del servicio de catálogo.

Antes de empezar

Debe instalar WebSphere Application Server y WebSphere eXtreme Scale.Si desea más información, consulte Capítulo 4, "Instalación y despliegue de WebSphere eXtreme Scale", en la página 27.

Por qué y cuándo se efectúa esta tarea

El proceso de servicio de catálogo se ejecuta en procesos de WebSphere Application Server. Cuando se inicia el proceso que contiene el proceso del servicio de catálogo, el proceso del servicio de catálogo también se inicia. Para el WebSphere Application Server básico, el servicio de catálogo se ejecuta en el proceso de

servidor. Para WebSphere Application Server Network Deployment, el proceso de servicio de catálogo se ejecuta automáticamente en el gestor de despliegue, aunque puede configurarse para que se ejecute también en un proceso de servidor de aplicaciones o agente de nodo.

- **Inicio de un único servidor de catálogo en el WebSphere Application Server base**

  Si WebSphere eXtreme Scale se ha instalado en WebSphere Application Server, el servicio de catálogo se inicia automáticamente en el proceso de servidor. Esta característica simplifica la prueba de unidades en entornos de despliegue, como por ejemplo Rational Application Developer porque no es necesario iniciar de forma explícita el servicio de catálogo.

  Si WebSphere eXtreme Scale se ha instalado en WebSphere Application Server Network Deployment, el servicio de catálogo se inicia automáticamente en el proceso del gestor de despliegue.

- **Inicio de una cuadrícula del servicio de catálogo en WebSphere Application Server Network Deployment**

  La cuadrícula del servidor de catálogo de eXtreme Scale se puede definir de forma explícita en un gestor de despliegue, agente de nodo o un proceso de servidor de aplicaciones utilizando la propiedad personalizada catalog.services.cluster, que se establece en la célula de WebSphere Application Server. Establezca el valor de la propiedad personalizada con el siguiente formato:*serverName*:*hostName*:*clientPort*:peerPort:*listenerPort*. Puede especificar varios servidores separando los valores con una coma.

  No los servicios de catálogo y los contenedores de eXtreme Scale no deben compartir ubicación en un entorno de producción. Incluya el servicio de catálogo en varios procesos de agente de nodo o en una cuadrícula de servidor de aplicaciones que no contenga ninguna aplicación eXtreme Scale. Con algunas excepciones, la propiedad personalizada catalog.services.cluster es similar al parámetro **-catalogServiceEndPoints** que se utiliza al iniciar un servidor de catálogo autónomo con la adición del parámetro **-listenerPort**. A continuación se definen cada uno de los atributos del valor de la propiedad personalizada:

  serverName
  : Especifica el nombre completo del proceso de WebSphere, como cellName, nodeName o serverName, del servidor que aloja el servicio de catálogo.

    Ejemplo: `cellA\node1\nodeagent`

  hostName
  : Especifica el nombre del servidor de alojamiento.

  clientPort
  : Especifica el puerto que se utiliza para la comunicación de la cuadrícula de catálogo de igual.

  peerPort
  : Especifica el puerto que se utiliza para la comunicación de la cuadrícula de catálogo de igual.

  listenerPort
  : El listenerPort debe coincidir con el valor BOOTSTRAP_ADDRESS que se define en la configuración del servidor WebSphere.

  Para crear la propiedad personalizada en la consola administrativa, pulse **Administración del sistema** → **Célula** → **Propiedades personalizadas** → **Nuevo**.

Especifique el nombre `catalog.services.cluster` y el valor en el formato adecuado utilizando los atributos definidos. Un ejemplo de un valor válido es:

```
cellA\node1\nodeagent:host.local.domain:6600:6601:2809,cellA\node2\
nodeagent:host.foreign.domain:6600:6601:2809
```

Después de establecer la propiedad personalizada, debe reiniciar cada servidor identificado. Cuando se reinician estos servidores, se inicia automáticamente la cuadrícula del servicio de catálogo.

Restricción: La cuadrícula del catálogo utiliza los mecanismos de grupo principal de WebSphere Application Server. Como resultado, los miembros de una cuadrícula de catálogo no amplían los límites de un grupo principal y, por lo tanto, una cuadrícula de catálogo no puede ampliar las células. Sin embargo, eXtreme Scale puede ampliar las células conectándose a un servidor de catálogo entre los límites de célula como, por ejemplo, una cuadrícula de catálogo autónomo o una cuadrícula de catálogo incorporada en otra célula.

Inicio de procesos de contenedor en un entorno deWebSphere Application Server

Los procesos de contenedor en un entorno WebSphere Application Server se inician automáticamente cuando se inicia un módulo que tiene incluidos los archivos XML de eXtreme Scale.

Antes de empezar

WebSphere Application Server y WebSphere eXtreme Scale deben estar instalados y debe poder acceder a la consola administrativa de WebSphere Application Server.

Por qué y cuándo se efectúa esta tarea

Las aplicaciones Java Platform, Enterprise Edition tienen reglas de cargador de clases complejas que complican enormemente la carga de clases cuando se utiliza un WebSphere eXtreme Scale compartido dentro de un servidor de Java EE. Normalmente, una aplicación Java EE es un único archivo EAR (Enterprise Archive) con uno o más módulos EJB (Enterprise JavaBeans™) o WAR (Web Archive) desplegados en la misma.

WebSphere eXtreme Scale observa con atención que cada módulo se inicie y busca los archivos XML de eXtreme Scale. Si WebSphere eXtreme Scale detecta que el módulo se inicia con los archivos XML, registra el servidor de aplicaciones como una máquina virtual Java (JVM) de contenedor de eXtreme Scale con el servicio de catálogo. Al registrar los contenedores con el servicio de catálogo, la misma aplicación se puede desplegar en distintas cuadrículas y seguir siendo tratados como una cuadrícula única por el servicio de catálogo. El servicio de catálogo no se ocupa de las células, cuadrículas o cuadrículas dinámicas. Todo es una contenedor de JVM de contenedor de eXtreme Scale o no lo es. Una única cuadrícula lógica puede abarcar varias células WebSphere Extended Deployment si es necesario.

1. Empaquete el archivo EAR de modo que tenga módulos que incluyan los archivos XML de eXtreme Scale en la carpeta META-INF. WebSphere eXtreme Scale detecta la presencia de los archivos `objectGrid.xml` y `objectGridDeployment.xml` en la carpeta META-INF de los módulos EJB y WEB cuando se inician. Si sólo se encuentra un archivo `objectGrid.xml`, se supone

que la JVM es un cliente, de lo contrario, se asume que esta JVM sirve de contenedor para la cuadrícula que se ha definido en el archivo `objectGridDeployment.xml`.

Debe utilizar los nombres correctos para estos archivos XML. Los nombres de archivo son sensibles a las mayúsculas y minúsculas. Si los archivos no existen, el contenedor no se inicia. Puede comprobar el archivo `systemout.log` para obtener mensajes que indican que se han colocado los fragmentos. Un módulo EJB o módulo WAR que utiliza eXtreme Scale debe tener archivos XML de eXtreme Scale en su directorio `META-INF`.

Los archivos XML de eXtreme Scale incluyen:

- Un archivo `objectGrid.xml`, conocido comúnmente como archivo XML de descriptor de eXtreme Scale.
- Un archivo `objectGridDeployment.xml`, conocido comúnmente como archivo XML de descriptor de despliegue.
- Un archivo `entity.xml` si se utilizan entidades (opcional). El nombre del archivo `entity.xml` debe coincidir con el nombre que se especifica en el archivo `objectGrid.xml`.

El tiempo de ejecución de eXtreme Scale detecta estos archivos y luego se pone en contacto con el servicio de catálogo para informar que hay otro contenedor disponible para alojar fragmentos para ese eXtreme Scale.

Consejo: Si piensa probar una aplicación con entidades que utilizan un servidor de eXtreme Scale, establezca el valor minSyncReplicas en `0` en el archivo XML de descriptor de despliegue para evitar que aparezca un mensaje `CWPRJ1005E: Error al resolver una excepción de asociación de entidades debido a que el repositorio EntityMetadata no está disponible. Se ha alcanzado el umbral de tiempo de espera`.

2. Despliegue e inicie la aplicación,

   El contenedor se inicia automáticamente cuando se inicia el módulo. El servicio de catálogo empieza a colocar primarios y réplicas (fragmentos) de particiones lo antes posible. Esta colocación se produce inmediatamente salvo que se defina un atributo numInitialContainers en el archivo `objectGridDeployment.xml`. Si define el atributo numInitialContainers, la colocación empieza cuando se ha iniciado esa cantidad de contenedores.

Qué hacer a continuación

Las aplicaciones que están en la misma célula que los contenedores pueden conectarse a estas cuadrículas utilizando un método ObjectGridManager.connect(null, null) y luego llamar al método getObjectGrid(ccc, "object grid name"). Los métodos connect o getObjectGrid pueden bloquearse hasta que los contenedores han terminado de colocar los fragmentos, aunque este bloqueo sólo es un problema cuando se inicia la cuadrícula.

Cargadores de clases

Todos los plug-ins u objetos almacenados en un eXtreme Scale se cargan en un cargador de clases determinado. Dos módulos EJB en el mismo archivo EAR pueden incluir estos objetos. Los objetos son los mismos pero se cargan utilizando distintos cargadores de clases. Si la aplicación A almacena un objeto Person en una correlación eXtreme Scale que es local respecto al servidor, la aplicación B recibe una ClassCastException si intenta leer dicho objeto. Esta excepción se produce porque la aplicación B ha cargado el objeto Person en un cargador de clases distinto.

Un método para resolver este problema es hacer que un módulo root contenga los plug-ins y objetos necesarios que están almacenados en el eXtreme Scale. Cada módulo que utiliza eXtreme Scale debe hacer referencia a dicho módulo para sus clases. Otra resolución es colocar estos objetos compartidos en un jar del programa de utilidad que esté en un cargador de clases común compartido tanto por módulos como por aplicaciones. Los objetos también se pueden colocar en las clases WebSphere o el directorio `lib/ext`, pero esto complica el despliegue y no es recomendable.

Los módulos EJB en un archivo EAR normalmente comparten el mismo ClassLoader y no se ven afectados por este problema. Cada módulo WAR tiene su propio ClassLoader y se ve afectado por este problema.

Conexión a una cuadrícula sólo como cliente

Si la propiedad catalog.services.cluster se ha definido en las propiedades personalizadas de célula, nodo o servidor, los módulos del archivo EAR pueden llamar al método ObjectGridManager.connect (ServerFactory.getServerProperties().getCatalogServiceBootstrap(), null, null) para obtener un ClientClusterContext y llamar al método ObjectGridManager.getObjectGrid(ccc, ″nombre cuadrícula″) para obtener una referencia a eXtreme Scale. Si todos los objetos de aplicación están almacenados en correlaciones, verifique que estos objetos están presentes en un ClassLoader común.

Los clientes de Java Platform, Standard Edition o los clientes fuera de la célula se pueden conectar utilizando el puerto IIOP del programa de arranque del servicio de catálogo (el valor predeterminado en Websphere es el gestor de despliegue) para obtener un ClientClusterContext y obtener después un eXtreme Scale determinado de la forma habitual.

Gestor de entidades

El gestor de entidades ayuda porque los tuples se almacenan en las correlaciones, no los objetos de correlación, de modo que hay menos problemas de cargador de clases. Sin embargo, los plug-ins pueden ser un problema. Tenga en cuenta también que un archivo XML de descriptor eXtreme Scale de alteración temporal de cliente siempre es necesario al conectarse a un eXtreme Scale que tiene entidades definidas: ObjectGridManager.connect(″host:port[,host:port], null, objectGridOverride) o ObjectGridManager.connect(null, objectGridOverride).

Configuración de los puertos TCP (protocolo de control de transmisiones) en un entorno de WebSphere Application Server

El servicio de catálogo ejecuta una instancia única dentro del gestor de despliegue de forma predeterminada y utiliza el puerto de programa de arranque del protocolo IIOP (Internet Inter-ORBProtocol) para el gestor de despliegue máquina virtual Java.

Por qué y cuándo se efectúa esta tarea

Las aplicaciones web o las aplicaciones EJB (Enterprise JavaBeans) dentro de una célula se pueden conectar a las cuadrículas que están dentro de la misma célula utilizado la llamada de conexión `null,null`, en lugar de especificar los puertos de programa de arranque del servicio de catálogo.

Si el gestor de despliegue aloja la cuadrícula del servicio de catálogo en WebSphere Application Server, los clientes externos de la célula (incluidos los clientes Java Platform, Standard Edition) se deben conectar al servicio de catálogo utilizando el nombre del sistema principal del gestor de despliegue y el puerto de programa de arranque de IIOP.

Cuando el servicio de catálogo se ejecuta en células de WebSphere Application Server mientras que los clientes se ejecutan fuera de las células, debe localizar los puertos de conexión de cliente localizando las propiedades personalizadas de la célula con el nombre `catalog.services.cluster`. Si se encuentra esta entrada, utilícela para conectar los clientes con el servicio del catálogo, pero si no se ha encontrado, utilice el puerto IIOP en el gestor de despliegue para la conexión de cliente. Si los clientes están en células de WebSphere Application Server, puede recuperar los puertos desde la interfaz CatalogServerProperties directamente.

eXtreme Scale reutiliza los puertos DCS (Distribution and Consistency Services) de High Availability Manager para la pertenencia a grupos. También se reutilizan los puertos JMX (Java Management Extensions).

Gestión de sesiones HTTP

En un entorno de WebSphere Application Server versión 5 o posterior, el gestor de sesiones HTTP que se distribuye con WebSphere eXtreme Scale puede alterar temporalmente el gestor de sesiones predeterminado del servidor de aplicaciones.

El gestor de sesiones HTTP de WebSphere eXtreme Scale también puede ejecutarse en WebSphere Application Server versión 6.0.2 o posterior, o en un entorno que no ejecute WebSphere Application Server, como por ejemplo WebSphere Application Server Community Edition o Apache Tomcat.

Características

El gestor de sesiones se ha diseñado de modo que pueda ejecutarse en cualquier contenedor Java Platform, Enterprise Edition versión 1.4. El gestor de sesiones no tiene dependencias en las API de WebSphere, por lo que puede admitir varias versiones de WebSphere Application Server y entornos de servidor de aplicaciones de proveedores.

El gestor de sesiones HTTP proporciona funciones de gestión de sesiones para una aplicación asociada. El gestor de sesiones crea sesiones HTTP y gestiona los ciclos de vida de dichas sesiones asociadas con la aplicación. Estas actividades de gestión de ciclo de vida incluyen: la invalidación de sesiones basadas en un tiempo de espera o una llamada a un servlet explícito o JavaServer Pages (JSP) y la invocación de escuchas de sesión asociados a la sesión o a la aplicación web. El gestor de sesiones persiste sus sesiones en una instancia de ObjectGrid. Esta instancia puede ser una instancia local, en memoria o una instancia completamente replicada, agrupada en clúster y particionada. El uso de esta última topología permite al gestor de sesiones proporcionar soporte de sustitución por anomalía de sesiones HTTP cuando los servidores de aplicaciones concluyen o se cierren de forma inesperada. El gestor de sesiones también puede funcionar en entornos que no admitan afinidad, si la afinidad no la aplica un nivel de equilibrador de carga que distribuya solicitudes al nivel de servidor de aplicaciones.

Escenarios de uso

El gestor de sesiones se puede utilizar en los siguientes escenarios:

- En entornos que utilizan servidores de aplicaciones en diferentes versiones de WebSphere Application Server, como por ejemplo en un escenario de migración clásico.
- En despliegues que utilizan servidores de aplicaciones de proveedores diferentes. Por ejemplo, una aplicación que se desarrolla en servidores de aplicaciones de código abierto y que se aloja en WebSphere Application Server. Otro ejemplo es una aplicación que se promociona de la fase intermedia a la de producción. Es posible realizar una migración sin interrupciones de estas versiones del servidor de aplicación mientras todas las sesiones HTTP están activas y dan servicio.
- En entornos que requieren que el usuario persista las sesiones con niveles superiores de calidad de servicio (QoS) y mejores garantías de disponibilidad de sesiones durante la sustitución por anomalía del servidor que los niveles Qos predeterminados de WebSphere Application Server.
- En un entorno donde la afinidad de sesiones no puede garantizarse, o en entornos en los que la afinidad se mantiene mediante un equilibrador de carga del proveedor y el mecanismo de afinidad debe personalizarse de acuerdo con dicho equilibrador de carga.
- En un entorno donde se descarga la sobrecarga de la gestión de sesiones y se almacena en un proceso Java externo.
- En varias células que permiten la sustitución por anomalía de las sesiones entre las células.

Cómo funciona el gestor de sesiones

El gestor de sesiones se inserta en la vía de acceso de la solicitud con el formato de un filtro de servlet. Puede añadir este filtro de servlet en todos los módulos web de la aplicación con las herramientas que se proporcionan con WebSphere eXtreme Scale. También puede añadir estos filtros de forma manual al descriptor de despliegue web de la aplicación. Este filtro recibe la solicitud antes que los archivos JSP o servlet en la aplicación de destino. En este momento, el filtro intercepta los objetos HTTPRequest y HTTPResponse y crea un objeto que los envuelve con su propia implementación.

La implementación de este filtro intercepta todas las llamadas relacionadas con la sesión HTTP que se realizan en los objetos HTTPRequest y HTTPResponse. El gestor de sesiones se encarga de estas llamadas, que no se pasan al gestor de sesiones base en la implementación del servidor Java Platform, Enterprise Edition subyacente. El gestor de sesiones crea y mantiene sesiones y los ciclos de vida de éstas, incluidas las invalidaciones basadas en el tiempo de espera y la activación de escuchas en las invalidaciones de sesiones y otros sucesos de ciclo de vida.

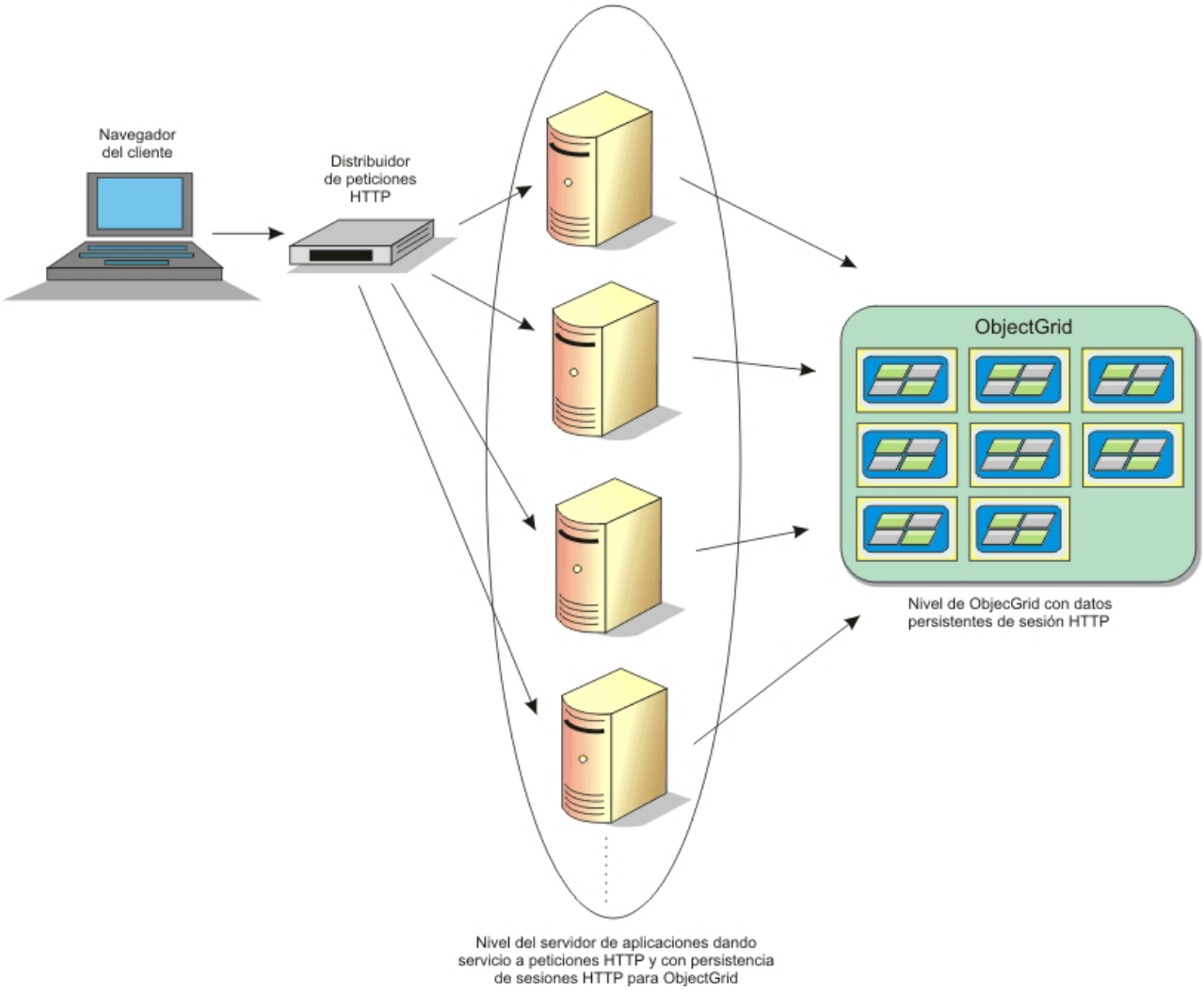


Figura 6. Topología de gestión de sesiones HTTP con una configuración de contenedor remoto

Topologías de despliegue

El gestor de sesiones puede configurarse mediante el uso de dos escenarios de despliegue dinámico diferentes:

- **Contenedores de eXtreme Scale incorporados, conectados a la red**

  En este escenario, los servidores eXtreme Scale se colocan en los mismos procesos que los servlets. El gestor de sesiones se puede comunicar directamente con la instancia de ObjectGrid local, lo que evita costosos retrasos de red.
- **Contenedores de eXtreme Scale remotos, conectados a la red**

  En este escenario, los servidores eXtreme Scale ejecutan en procesos externos el proceso en el que se ejecutan los servlets. El gestor de sesiones se comunica con un eXtreme Scale remoto.

Recuperación de una sesión de eXtreme Scale en una aplicación de gestor de sesiones

```
public void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {

        HttpSession session = req.getSession(true);

        System.out.println("calling getSession");
```

```
    // llamar getAttribute("com.ibm.websphere.objectgrid.session")
// para obtener la instancia de la sesión ObjectGrid
    Session ogSession = (Session)session.getAttribute
  ("com.ibm.websphere.objectgrid.session");

    System.out.println("ogSession = "+ogSession);
}
```

Los cambios realizados en las correlaciones con la sesión que se devuelve de la llamada al método getAttribute se confirman cuando se confirma la sesión subyacente.

Tareas relacionadas

"Configuración del gestor de sesiones de WebSphere eXtreme Scale para que funcione con WebSphere Application Server"
Mientras WebSphere Application Server proporciona una función de gestión de sesiones, este soporte no se redimensiona bajo cargas de solicitudes extremas. WebSphere eXtreme Scale se entrega empaquetado con una implementación de gestión de sesiones que altera temporalmente el gestor de sesiones predeterminado para un contenedor web y proporciona una mejor escalabilidad y opciones de configuración más potentes.

"Configuración del gestor de sesiones HTTP para trabajar con WebSphere Application Server Community Edition" en la página 256
WebSphere Application Server Community Edition puede compartir el estado de sesión, pero no de una forma eficaz y escalable. WebSphere eXtreme Scale proporciona un alto rendimiento, una capa de persistencia distribuida que puede utilizarse para replicar el estado, pero que no se integra fácilmente con otro servidor de aplicaciones fuera de WebSphere Application Server. Puede integrar estos dos productos para proporcionar una solución de gestión de sesiones escalable. Puede utilizar la infraestructura modular de WebSphere Application Server Community Edition, GBeans, para incluir WebSphere eXtreme Scale como mecanismo de persistencia de estado de sesión.

Referencia relacionada

"Parámetros de inicialización del contexto del servlet" en la página 253
La siguiente lista de parámetros de inicialización de contexto de servlet se puede especifica en el archivo de propiedades tal como es necesario en el script o los métodos de unión basado en ANT.

Configuración del gestor de sesiones de WebSphere eXtreme Scale para que funcione con WebSphere Application Server

Mientras WebSphere Application Server proporciona una función de gestión de sesiones, este soporte no se redimensiona bajo cargas de solicitudes extremas. WebSphere eXtreme Scale se entrega empaquetado con una implementación de gestión de sesiones que altera temporalmente el gestor de sesiones predeterminado para un contenedor web y proporciona una mejor escalabilidad y opciones de configuración más potentes.

Por qué y cuándo se efectúa esta tarea

El gestor de sesiones HTTP de WebSphere eXtreme Scale soporta ambos servidores, los remotos y los incrustados, para el almacenamiento en la memoria caché.

Caso de ejemplo de incrustado

En el caso de ejemplo de incrustado, los servidores de WebSphere eXtreme Scale comparten ubicación en los mismos procesos donde se ejecutan los servlets. El

gestor de sesiones se puede comunicar directamente con la instancia local de ObjectGrid, lo que evita costosos retardos de red.

Si utiliza WebSphere Application Server, coloque los archivos `objectgridRoot/session/samples/objectGrid.xml` y `objectgridRoot/session/samples/objectGridDeployment.xml` proporcionados en los directorios META-INF de los archivos WAR (Web Archive). eXtreme Scale detecta automáticamente estos archivos cuando se inicia la aplicación e inicia automáticamente los contenedores de eXtreme Scale en el mismo proceso que el gestor de sesiones.

Puede modificar el archivo `objectGridDeployment.xml` en función de si desea utilizar la réplica síncrona o asíncrona y de cuantas réplicas desea configurar.

Caso de ejemplo de servidores remotos

En el escenario de servidores remotos, los servidores eXtreme Scale se ejecutan en distintos procesos que los servlets. El gestor de sesiones se comunica con un servidor de eXtreme Scale remoto. Para utilizar un servidor de eXtreme Scale remoto conectado a la red, el gestor de sesiones se debe configurar con los nombres de host y los números de puertos del clúster de servicio de catálogo de eXtreme Scale. El gestor de sesiones utilizará una conexión de cliente de eXtreme Scale para comunicarse con el servidor de catálogo y los servidores de eXtreme Scale.

Si los servidores eXtreme Scale se van a iniciar en procesos independientes y autónomos, inicie los contenedores eXtreme Scale utilizando los archivos `objectGridStandAlone.xml` y `objectGridDeploymentStandAlone.xml` proporcionados en el directorio de ejemplos del gestor de sesiones.

1. **Si ejecuta WebSphere Application Server oWebSphere Application Server Network Deployment sin una instalación de WebSphere eXtreme Scale:** instale las bibliotecas de eXtreme Scale.

   Copie los archivos `objectgridRoot/session/lib/sessionobjectgrid.jar` y `objectgridRoot/lib/wsobjectgrid.jar` en la classpath de los servidores de aplicaciones que incluyen la nueva aplicación habilitada para la gestión de sesiones HTTP. Coloque estos archivos JAR (Java Archive) en la carpeta `<WAS_HOME>/lib` y reinicie el servidor para que sean visibles estas clases.
2. Una la aplicación de modo que pueda utilizar el gestor de sesiones. Para utilizar el gestor de sesiones, debe añadir las declaraciones de filtro apropiadas a los descriptores de despliegue web para la aplicación. Además, los parámetros de configuración del gestor de sesiones se pasan al gestor de sesiones en el formato de parámetros de inicialización de contexto de servlet en los descriptores de despliegue. Existen tres formas en las que puede introducir esta información en la aplicación:
   - **Script addObjectGridFilter**

     Utilice un script de línea de mandatos proporcionado junto con eXtreme Scale para unir una aplicación con declaraciones de filtro y la configuración en el formato de parámetros de inicialización de contexto de servlet. Este script, `objectgridRoot/bin/addObjectGridFilter.sh` o`objectgridRoot/bin/addObjectGridFilter.bat`, acepta dos parámetros: la aplicación (vía de acceso absoluta para el archivo de archivador empresarial) que es necesario unir y la vía de acceso absoluta al archivo de propiedades que contiene varias propiedades de configuración. El formato de uso de este script es el siguiente: Windows

     ```
     addObjectGridFilter.bat [ubicación archivo ear] [ubicación archivo propiedades]
     ```

UNIX

```
addObjectGridFilter.sh [ubicación archivo ear] [ubicación archivo propiedades]
```

UNIX **Ejemplo del uso de eXtreme Scale instalado en WebSphere Application Server en Unix:**

a. `cd objectgridRoot/optionalLibraries/ObjectGrid/session/bin`

b. `addObjectGridFilter.sh /tmp/mySessionTest.ear wasRoot/optionalLibraries/ObjectGrid/session/samples/splicer.properties`

UNIX **Ejemplo del uso de eXtreme Scale instalado en un directorio autónomo en Unix:**

a. `cd objectgridRoot/session/bin`

b. `addObjectGridFilter.sh /tmp/mySessionTest.ear objectgridRoot/session/samples/splicer.properties`

El filtro de servlet que se une mantiene los valores predeterminados para los valores de configuración. Puede alterar temporalmente estos valores predeterminados con las opciones de configuración que especifique en el archivo de propiedades en el segundo argumento. Para obtener una lista de los parámetros que puede utilizar, consulte "Parámetros de inicialización del contexto del servlet" en la página 253.

Puede modificar y utilizar el archivo `splicer.properties` de ejemplo proporcionado con la instalación de eXtreme Scale. También puede utilizar el script addObjectGridServlets, que inserta el gestor de sesiones ampliando cada servlet. No obstante, el script recomendado es el script addObjectGridFilter.

- **Script de compilación Ant**

  WebSphere eXtreme Scale se entrega con un archivo `build.xml` que puede utilizar Apache Ant, que se incluye en la carpeta `wasRoot/bin` de una instalación de WebSphere Application Server. El archivo `build.xml` puede modificarse para cambiar las propiedades de configuración del gestor de sesiones. Las propiedades de configuración son idénticas a los nombres de propiedades del archivo `splicer.properties`. Después de modificar el archivo `build.xml`, se invoca el proceso Ant ejecutando ant.sh, ws_ant.sh (UNIX) o ant.bat, ws_ant.bat (Windows).

- **Actualizar manualmente el descriptor web**

  Edite el archivo `web.xml` que se empaqueta con la aplicación web para incorporar la declaración de filtro, su correlación de servlets y los parámetros de inicialización de contexto de servlet. No debe utilizar este método porque es propenso a errores.

Para obtener una lista de los parámetros que puede utilizar, consulte "Parámetros de inicialización del contexto del servlet" en la página 253.

3. Despliegue la aplicación. Despliegue la aplicación con un conjunto de pasos normales para un servidor o un clúster. Después de desplegar la aplicación, puede iniciarla.
4. Acceda a la aplicación. Ahora puede acceder a la aplicación, que interactúa con el gestor de sesiones y WebSphere eXtreme Scale.

Qué hacer a continuación

Puede cambiar la mayoría de los atributos de configuración para el gestor de sesiones cuando indica a la aplicación que utilice el gestor de sesiones. Estos atributos incluyen variaciones en el tipo de réplica (síncrona o asíncrona), la

longitud del ID de sesión, el tamaño de la tabla de sesión en memoria, etc. Aparte de los atributos que pueden cambiarse durante la instrumentación de la aplicación, los otros únicos atributos de configuración que puede cambiar después del despliegue de la aplicación son los atributos relacionados con la topología de clúster de servidores de WebSphere eXtreme Scale y la forma en que sus clientes (gestores de sesiones) se conectan a los mismos.

Conceptos relacionados

"Gestión de sesiones HTTP" en la página 247
En un entorno de WebSphere Application Server versión 5 o posterior, el gestor de sesiones HTTP que se distribuye con WebSphere eXtreme Scale puede alterar temporalmente el gestor de sesiones predeterminado del servidor de aplicaciones.

Referencia relacionada

"Parámetros de inicialización del contexto del servlet"
La siguiente lista de parámetros de inicialización de contexto de servlet se puede especifica en el archivo de propiedades tal como es necesario en el script o los métodos de unión basado en ANT.

Parámetros de inicialización del contexto del servlet

La siguiente lista de parámetros de inicialización de contexto de servlet se puede especifica en el archivo de propiedades tal como es necesario en el script o los métodos de unión basado en ANT.

Parámetros

affinityManager
Especifica el paquete plenamente calificado y el nombre de clase Java de un plug-in de filtrado de servlets para facilitar la afinidad de direccionamiento de sesiones HTTP. Este valor se ignora para WebSphere Application Server porque el soporte de afinidad está incorporado.Indique uno de los siguientes valores:

- **Sin afinidad (valor predeterminado):** com.ibm.ws.httpsession.NoAffinityManager
- **Mecanismo de afinidad de proveedor:** com.ibm.ws.httpsession.AssumeAffinityManager
- **Construir un nuevo gestor de afinidad:** utilice la interfaz com.ibm.wsspi.session.ISessionAffinityManager

persistenceMechanism
Especifica un valor de serie que define cómo se almacena la sesión en eXtreme Scale. Indique uno de los siguientes valores:

- **ObjectGridStore**: cada atributo de sesión se almacena como una entrada distinta en la tabla de eXtreme Scale.
- **ObjectGridAtomicSessionStore**: toda la sesión se almacena como una entrada única en la tabla de eXtreme Scale.

objectGridName
Especifica un valor de serie que define el nombre de la tabla que se utiliza para una aplicación web concreta. El nombre predeterminado se toma del nombre de la aplicación web que se deriva del valor ServletContext. Si establece este parámetro se altera temporalmente ese valor.

catalogHostPort
Especifica la información de conexión del servidor de catálogo; es necesario que el valor tenga el formato `host:puerto<,host:puerto>`. Esta lista puede ser

arbitrariamente larga y se utiliza la primera dirección viable. Esta propiedad sólo es necesaria para el caso de ejemplo de eXtreme Scaleconectado a la red remoto.

replicationType
Un valor de serie que define cómo se escriben las actualizaciones a sesiones en eXtreme Scale. Los valores válidos son `asynchronous` y `synchronous`. El valor predeterminado es `asynchronous`, que sólo se puede aplicar si se utiliza la afinidad. De lo contrario, sólo se da soporte al valor synchronous.

replicationInterval
Un valor entero que define el intervalo en segundos entre la grabación de sesiones actualizadas para eXtreme Scale cuando el valor de parámetro replicationType es `asynchronous`. El valor predeterminado es 10 segundos.

sessionTableSize
Un valor entero que define el número de sesiones que se mantienen en memoria con el filtrado de servlets además de almacenarse en eXtreme Scale. El valor predeterminado es 1000.

defaultSessionTimeout
Un valor entero que define el intervalo de tiempo en minutos que una sesión puede estar inactiva (por ejemplo, que no se pueda acceder a la misma desde un servidor) antes de anular y eliminar la sesión del sistema. El valor predeterminado es de 30 minutos.

sessionIDLength
Un valor entero que define la longitud de los identificadores de serie que se crean para sesiones HTTP. El valor predeterminado es 23.

shareSessionsAcrossWebApps
Especifica un valor de serie de true o false. El valor predeterminado es false. Según la especificación de servlet, las sesiones HTTP no pueden compartirse entre las aplicaciones web. Para permitir que se puedan compartir, se proporciona una extensión para la especificación del servlet.

cookieName
Especifica un valor de serie que define el nombre de la cookie esta aplicación web. El valor predeterminado es JSESSIONID. Si desea utilizar un nombre de cookie exclusivo, añada esta propiedad al archivo `splicer.properties`.

Conceptos relacionados

"Gestión de sesiones HTTP" en la página 247
En un entorno de WebSphere Application Server versión 5 o posterior, el gestor de sesiones HTTP que se distribuye con WebSphere eXtreme Scale puede alterar temporalmente el gestor de sesiones predeterminado del servidor de aplicaciones.

Tareas relacionadas

"Configuración del gestor de sesiones de WebSphere eXtreme Scale para que funcione con WebSphere Application Server" en la página 250
Mientras WebSphere Application Server proporciona una función de gestión de sesiones, este soporte no se redimensiona bajo cargas de solicitudes extremas. WebSphere eXtreme Scale se entrega empaquetado con una implementación de gestión de sesiones que altera temporalmente el gestor de sesiones predeterminado para un contenedor web y proporciona una mejor escalabilidad y opciones de configuración más potentes.

"Configuración del gestor de sesiones HTTP para trabajar con WebSphere Application Server Community Edition" en la página 256
WebSphere Application Server Community Edition puede compartir el estado de sesión, pero no de una forma eficaz y escalable. WebSphere eXtreme Scale proporciona un alto rendimiento, una capa de persistencia distribuida que puede utilizarse para replicar el estado, pero que no se integra fácilmente con otro servidor de aplicaciones fuera de WebSphere Application Server. Puede integrar estos dos productos para proporcionar una solución de gestión de sesiones escalable. Puede utilizar la infraestructura modular de WebSphere Application Server Community Edition, GBeans, para incluir WebSphere eXtreme Scale como mecanismo de persistencia de estado de sesión.

Uso de WebSphere eXtreme Scale para la gestión de sesiones SIP

Puede utilizar WebSphere eXtreme Scale como un mecanismo de réplica SIP (Session Initiation Protocol) como alternativa fiable para el servicio de duplicación de datos (DRS) para la réplica de sesiones SIP.

Configuración de la gestión de sesiones SIP

Para utilizar WebSphere eXtreme Scale como el mecanismo de réplica SIP, establezca la propiedad personalizada com.ibm.sip.ha.replicator.type. En la consola de administración, seleccione **Servidores de aplicaciones** → ***mi_servidor_aplicaciones*** → **Contenedor SIP** → **Propiedades personalizadas** para cada servidor para añadir la propiedad personalizada. Escriba `com.ibm.sip.ha.replicator.type` como nombre (Name) y `OBJECTGRID` como valor (Value).

Utilice las propiedades siguientes para personalizar el comportamiento del objeto ObjectGrid que se usa para almacenar sesiones SIP. En la consola de administración, pulse **Servidores de aplicaciones** → ***mi_servidor_aplicaciones*** → **Contenedor SIP** → **Propiedades personalizadas** para cada servidor para añadir la propiedad personalizada. Escriba el **Nombre** y el **Valor**. Cada servidor debe tener el mismo conjunto de propiedades para funcionar correctamente.

Tabla 8. Propiedades personalizadas para la gestión de sesiones SIP con ObjectGrid

Propiedad	Valor	Valor predeterminado
com.ibm.sip.ha.replicator.type	`OBJECTGRID`: utilizar ObjectGrid como un almacén de sesiones SIP	

Tabla 8. Propiedades personalizadas para la gestión de sesiones SIP con ObjectGrid (continuación)

Propiedad	Valor	Valor predeterminado
min.synchronous.replicas	Número mínimo de réplicas síncronas	0
max.synchronous.replicas	Número máximo de réplicas síncronas	0
max.asynchronous.replicas	Número máximo de réplicas asíncronas	1
auto.replace.lost.shards	Si desea más información, consulte "Configuración de topología de despliegue" en la página 149.	true
development.mode	• `true` - permite que las réplicas estén activas en el mismo nodo que los primarios • `false` - las réplicas deben estar en nodos distintos que los primarios	false

Configuración del gestor de sesiones HTTP para trabajar con WebSphere Application Server Community Edition

WebSphere Application Server Community Edition puede compartir el estado de sesión, pero no de una forma eficaz y escalable. WebSphere eXtreme Scale proporciona un alto rendimiento, una capa de persistencia distribuida que puede utilizarse para replicar el estado, pero que no se integra fácilmente con otro servidor de aplicaciones fuera de WebSphere Application Server. Puede integrar estos dos productos para proporcionar una solución de gestión de sesiones escalable. Puede utilizar la infraestructura modular de WebSphere Application Server Community Edition, GBeans, para incluir WebSphere eXtreme Scale como mecanismo de persistencia de estado de sesión.

Antes de empezar

Debe descomprimir el archivo zip o instalar Geronimo o WebSphere Application Server Community Edition y WebSphere eXtreme Scale en el sistema.

Por qué y cuándo se efectúa esta tarea

El eje de WebSphere Application Server Community Edition, el kernel, depende de módulos externos, denominados GBeans, para proporcionar funciones y prestaciones.

1. Distribuya los GBeans.
   a. Vaya hasta el directorio *raíz_extremeScale*/wasce/bin.
   b. Ejecute el script addToCeRepository para el sistema que está ejecutando.
      - Linux UNIX addToRepository.sh ceRoot
      - Windows addToRepository.bat ceRoot
2. Edite el archivo de configuración, Ahora que el GBean se ha distribuido, configure WebSphere Application Server Community Edition para indicar que debe iniciarse durante el inicio del servidor. Puede registrar el GBean y suministrarlo con argumentos configurables que se pueden utilizar durante la ejecución con el archivo `config.xml`.
   a. Vaya al directorio `WasCeRoot/var/config`.
   b. Abra el archivo `config.xml` en un editor de texto plano.
   c. Busque la siguiente línea en el archivo `config.xml`:

```
<module name="org.apache.geronimo.plugins/mconsole-ds/2.1.1/car"/>
 </attributes>
```

d. Añada el siguiente fragmento de código antes del código </attributes>.

```
<module name="com.ibm.websphere.objectgrid/gbean/1.0/car">
   <gbean name="objectgrid/BringupPlugin">
   <attribute name="objectgridHome">c:\objectgrid</attribute>
   <attribute name="geronimoHome">C:\websphereCE</attribute>
   <attribute name="serverName">server1</attribute>
   <attribute name="catalogServiceEndPoints">host:port</attribute>
   <attribute name="replicationDisabled">false</attribute>
   <attribute name="traceSpecification">*=all=disabled</attribute>
   </gbean>
</module>
```

Los valores de los parámetros GBean en el fragmento de código anterior son ejemplos. Puede configurar estos valores.

objectgridHome
Especifica el directorio de instalación de objectgridRoot.

geronimoHome -
Especifica el directorio de instalación de WebSphere Application Server Community Edition o Apache Geronimo.

serverName
Especifica un nombre de servidor que debe ser exclusivo para cada instancia de servidor de WebSphere Application Server Community Edition yeXtreme Scale

catalogServiceEndPoints -
Especifica el host y el puerto del servidor de catálogo final.

replicationDisabled -
Especifica si la réplica del estado de sesión está habilitada.

traceSpecification -
Especifique una serie para el rastreo del contenedor eXtreme Scale dentro de la máquina virtual Java (JVM).

3. Inicie un servidor de catálogo. La imagen de WebSphere Application Server Community Edition puede iniciar satisfactoriamente un servidor eXtreme Scale dentro de su JVM durante el inicio del servidor. Sin embargo, para un programa de arranque de eXtreme Scale primero debe iniciar un servidor de catálogo. El servidor de catálogo es parecido a un gestor de despliegue dentro de una topología de WebSphere Application Server y debe iniciarse antes de que se inicie WebSphere Application Server Community Edition. Después de iniciar el servidor de catálogo, escriba el nombre de host y el puerto para el servidor de catálogo en el archivo `config.xml` dentro de la entrada de GBean. Si desea más información, consulte “Inicio del servicio de catálogo en un entorno autónomo” en la página 229. Asegúrese de leer la sección que trata sobre el enlace del puerto ORB con un nombre de host y puerto concreto. Es necesario especificar estos puntos finales dentro de la configuración de GBean. Si hay un conflicto de puertos durante el inicio del servidor, debe abrir el archivo `ceRoot/var/config/config-substitutions.properties` y cambiar la clave NamingPort por un valor distinto de `1099`.
4. Inicie WebSphere Application Server Community Edition para probar el programa de arranque.
   a. Vaya al directorio `WasCeRoot/bin`.
   b. Establezca la variable de entorno JAVA_HOME.
      - UNIX Linux `export JAVA_HOME=javaHome`
      - Windows `set JAVA_HOME=javaHome`

c. Ejecute el mandato de inicio.

- UNIX Linux `geronimo.sh run`
- Windows `geronimo.bat run`

El servidor debe iniciarse ahora y empezar a inicializar sus componentes o GBeans. El plug-in eXtreme Scale es el último plug-in en cargarse y muestra las sentencias de información y rastreo de inicio necesarias.

Conceptos relacionados

"Gestión de sesiones HTTP" en la página 247
En un entorno de WebSphere Application Server versión 5 o posterior, el gestor de sesiones HTTP que se distribuye con WebSphere eXtreme Scale puede alterar temporalmente el gestor de sesiones predeterminado del servidor de aplicaciones.

Referencia relacionada

"Parámetros de inicialización del contexto del servlet" en la página 253
La siguiente lista de parámetros de inicialización de contexto de servlet se puede especifica en el archivo de propiedades tal como es necesario en el script o los métodos de unión basado en ANT.

Utilización de WebSphere eXtreme Scale para la réplica del estado de sesión en WebSphere Application Server Community Edition

Cada aplicación web desplegada en WebSphere Application Server Community Edition requiere ser compatible con la especificación de Java Platform, Enterprise Edition, junto con el archivo descriptor `geronimo-web.xml`. Este archivo contiene información de acceso y dependencia específica al entorno de ejecución para WebSphere Application Server Community Edition. Para permitir a una aplicación web de Java EE utilizar la característica de réplica de sesión de WebSphere eXtreme Scale, la aplicación web debe proporcionar datos sobre el plug-in de eXtreme Scale como atributos específicos de sesión.

Antes de empezar

Este proceso es lo suficientemente sencillo para crearse en cualquier sistema operativo, sin IDE ni programas de utilidad.

Por qué y cuándo se efectúa esta tarea

Configure una aplicación web básica que utilice la persistencia de sesión más adelante en eXtreme Scale, que contabiliza y almacena el número de veces que un usuario ha accedido a un servlet determinado.

1. Cree la aplicación web de ejemplo.
   a. Cree un directorio en alguna ubicación arbitraria dentro del sistema de archivos. Por ejemplo, es posible que cree un directorio denominado *app_home*.
   b. Vaya hasta el directorio *app_home* y cree la siguiente estructura de directorio:

```
> app_home
--> META-INF
--> WEB-INF
-----> lib
-----> classes
```

c. Desde el directorio *app_home/WEB-INF*, cree un archivo `web.xml`. Copie y pegue el código siguiente en el archivo `web.xml`. Asegúrese de que todos los parámetros de contexto se añaden al archivo `web.xml`, o es posible que el filtro no funcione correctamente.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="sample" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
 <display-name>
 sample</display-name>
 <context-param>
    <param-name>shareSessionsAcrossWebApps</param-name>
    <param-value>false</param-value>
 </context-param>
 <context-param>
     <param-name>sessionIDLength</param-name>
     <param-value>23</param-value>
 </context-param>
 <context-param>
     <param-name>sessionTableSize</param-name>
     <param-value>1000</param-value>
 </context-param>
 <context-param>
     <param-name>replicationInterval</param-name>
     <param-value>10</param-value>
 </context-param>
 <context-param>
     <param-name>replicationType</param-name>
     <param-value>synchronous</param-value>
 </context-param>
 <context-param>
     <param-name>defaultSessionTimeout</param-name>
     <param-value>30</param-value>
 </context-param>
 <context-param>
     <param-name>affinityManager</param-name>
     <param-value>com.ibm.ws.httpsession.NoAffinityManager</param-value>
 </context-param>
 <context-param>
     <param-name>useURLEncoding</param-name>
     <param-value>true</param-value>
 </context-param>
 <context-param>
     <param-name>objectGridName</param-name>
     <param-value>session</param-value>
 </context-param>
 <context-param>
     <param-name>persistenceMechanism</param-name>
     <param-value>ObjectGridStore</param-value>
 </context-param>
 <filter>
     <filter-name>HttpSessionFilter</filter-name>
     <filter-class>com.ibm.ws.httpsession.HttpSessionFilter</filter-class>
 </filter>
 <filter-mapping>
     <filter-name>HttpSessionFilter</filter-name>
     <url-pattern>/*</url-pattern>
 </filter-mapping>
<servlet>
 <description>
 </description>
 <display-name>
 SampleServlet</display-name>
 <servlet-name>SampleServlet</servlet-name>
 <servlet-class>
 SampleServlet</servlet-class>
</servlet>
<servlet-mapping>
 <servlet-name>SampleServlet</servlet-name>
 <url-pattern>/SampleServlet</url-pattern>
</servlet-mapping>
<welcome-file-list>
 <welcome-file>index.html</welcome-file>
 <welcome-file>index.htm</welcome-file>
 <welcome-file>index.jsp</welcome-file>
 <welcome-file>default.html</welcome-file>
 <welcome-file>default.htm</welcome-file>
 <welcome-file>default.jsp</welcome-file>
</welcome-file-list>
</web-app>
```

d. Desde el directorio *app_home/WEB-INF*, cree un archivo `geronimo-web.xml`. Copie y pegue el código siguiente en el archivo `geronimo-web.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.1"
 xmlns:nam="http://geronimo.apache.org/xml/ns/naming-1.1"
 xmlns:sec="http://geronimo.apache.org/xml/ns/security-1.1"
 xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.1">
 <sys:environment>
  <sys:moduleId>
   <sys:groupId>com.ibm.websphere.objectgrid</sys:groupId>
   <sys:artifactId>sample</sys:artifactId>
   <sys:version>1.0</sys:version>
   <sys:type>war</sys:type>
  </sys:moduleId>
  <sys:dependencies>
   <sys:dependency>
    <sys:groupId>com.ibm.websphere.objectgrid
    </sys:groupId>
    <sys:artifactId>session</sys:artifactId>
    <sys:version>1.0</sys:version>
    <sys:type>jar</sys:type>
   </sys:dependency>
  </sys:dependencies>
 </sys:environment>
 <context-root>/sample</context-root>
</web-app>
```

e. Coloque el archivo `SampleServlet.class` conectado debajo del directorio *`WEB-INF/classes`*.

f. Vaya hasta el directorio *`app_home`*.

g. Ejecute el siguiente mandato para empaquetar el archivador web (WAR):

```
jar -cf sample.war
```

h. La aplicación web ahora está preparada para ser desplegada.

2. Despliegue la aplicación web.

a. Inicie una sesión en la página inicial de WebSphere Application Server Community Edition en el `nombre_host:puerto` especificado. De forma predeterminada, la página inicial es: `http://localhost:8080/`.

b. Pulse **Consola administrativa**.

c. Escriba el nombre de usuario y la contraseña y, a continuación, pulse **Aceptar**. De forma predeterminada, las credenciales son:

- nombre de usuario: System
- contraseña: Manager

d. Seleccione **Aplicaciones** → **Desplegar nuevas** en el menú de la izquierda.

e. Pulse **Examinar** a la derecha de Archivar y, a continuación, seleccione el archivo `sample.war` que ha creado en la sección anterior.

f. Deje en blanco **Plan**, porque ha empaquetado el plan con la aplicación web.

g. Seleccione **Iniciar aplicación después de instalar** y, a continuación, pulse **Instalar**.

Después de que se instale la aplicación, se visualiza un mensaje que indica que la instalación se ha realizado correctamente.

3. Pruebe la aplicación.

a. Vaya hasta la raíz de contexto `/sample/SampleServlet` para probar la aplicación. De forma predeterminada, la raíz de contexto es: `http://localhost:8080/sample/SampleServlet`.

b. Debe ver el número de veces que ha visitado este servlet, seguido por la implementación de HttpSession. Debe aparecer como HttpSessionImpl, si utiliza el nivel de persistencia de eXtreme Scale, o StandardSessionFacade si utiliza el mecanismo de almacenamiento-sesión predeterminado del contenedor web.

Capítulo 8. Supervisión del entorno de despliegue

Puede utilizar API, MBeans, registros y programas de utilidad para supervisar el rendimiento del entorno de aplicación.

Conceptos relacionados

"Visión general de las estadísticas"
Las estadísticas de WebSphere eXtreme Scale se calculan basándose en un árbol de estadísticas internas. La API StatsAccessor, los módulos PMI (Performance Monitoring Infrastructure) y la API MBean se generan desde el árbol interno.

"Herramientas del proveedor" en la página 290
WebSphere eXtreme Scale se puede supervisar utilizando varias soluciones populares de supervisión empresarial. Los agentes de plug-in se incluyen para IBM Tivoli Monitoring e Hyperic HQ, que supervisan WebSphere eXtreme Scale utilizando los beans de gestión a los que se puede acceder públicamente. CA Wily Introscope utiliza la instrumentación de métodos Java para capturar las estadísticas.

Referencia relacionada

"Utilización de beans gestionados (MBeans) para administrar el entorno" en la página 286
Puede utilizar varios tipos distintos de MBeans JMX (Java Management Extensions) para administrar y supervisar despliegues. Cada MBean hace referencia a una entidad específica como, por ejemplo, una correlación, eXtreme Scale, servidor, grupo de réplicas o miembro del grupo de réplicas.

Visión general de las estadísticas

Las estadísticas de WebSphere eXtreme Scale se calculan basándose en un árbol de estadísticas internas. La API StatsAccessor, los módulos PMI (Performance Monitoring Infrastructure) y la API MBean se generan desde el árbol interno.

La siguiente figura muestra la configuración general de las estadísticas para eXtreme Scale.

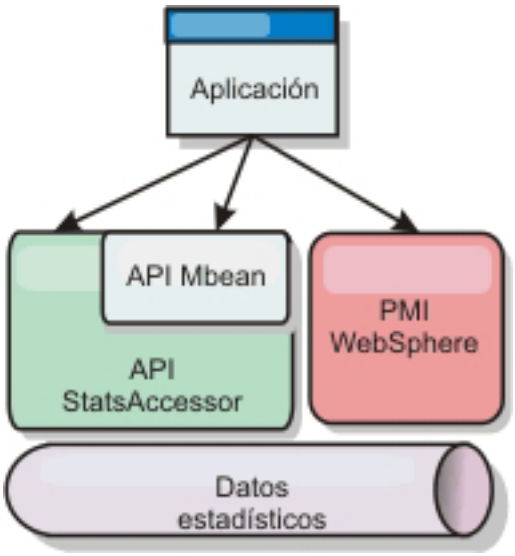


Figura 7. Visión general de las estadísticas

Cada una de estas API ofrecen una visión del árbol de estadísticas, pero se utilizan por distintos motivos:

- **API de estadísticas**: utilice la API de estadísticas como mecanismo de búsqueda genérico para los datos en el árbol de estadísticas internas. Puede utilizar la API de estadísticas para los clientes o las aplicaciones incorporadas. No puede utilizar la API de estadísticas si tiene un eXtreme Scale distribuido.
- **MBean API**: la API de MBean es un mecanismo basado en especificaciones para realizar la supervisión. La API MBean utiliza la API Statistics y se ejecuta de forma local en la máquina virtual Java (JVM) del servidor. Las estructuras de API y MBean se han diseñado para integrarse fácilmente con otros programas de utilidad. Utilice la API de MBean cuando ejecute un eXtreme Scale distribuido.
- **Módulos PMI (Performance Monitoring Infrastructure) de WebSphere Application Server**: utilice PMI si está ejecutando WebSphere eXtreme Scale dentro de WebSphere Application Server. Estos módulos proporcionan una vista del árbol de estadísticas internas.

API de estadísticas

De forma muy parecida a una correlación de árbol, hay una correspondiente vía de acceso y clave que se utiliza para recuperar un módulo específico, o en este caso el nivel de agregación o granularidad. Por ejemplo, suponga que siempre hay un nodo raíz arbitrario en el árbol y que las estadísticas se están recopilando para una correlación llamada ″payroll,″ que pertenece a un ObjectGrid denominado ″accounting.″ Por ejemplo, para acceder al módulo para el nivel de agregación o granularidad de una correlación, podría pasar una String[] de las vías de acceso. En este caso equivaldría a String[] {root, ″accounting″, ″payroll″}, ya que cada String representaría la vía de acceso del nodo. La ventaja de esta estructura es que un usuario puede especificar la matriz para cualquier código en la vía de acceso y obtener el nivel de agregación para ese nodo. Por lo tanto, si pasa String[] {root, ″accounting″} le proporcionará estadísticas de correlaciones, pero para toda la cuadrícula de ″accounting.″ Esto ofrece al usuario la capacidad de especificar tipos de estadísticas para supervisar y que nivel de agregación es necesario para la aplicación.

Módulos PMI de WebSphere Application Server

WebSphere eXtreme Scale incluye módulos de estadísticas para utilizarlos con la PMI de WebSphere Application Server. Cuando se aumenta un perfil de WebSphere Application Server con WebSphere eXtreme Scale, los scripts de aumento integran automáticamente los módulos de WebSphere eXtreme Scale en los archivos de configuración de WebSphere Application Server. Con PMI, puede habilitar e inhabilitar módulos de estadísticas, agregar estadísticas automáticamente en distinta granularidad, e incluso trazar un gráfico con los datos utilizando el Tivoli Performance Viewer incorporado. Si desea más información, consulte "Supervisión del rendimiento con PMI de WebSphere Application Server" en la página 270.

Integración de productos de proveedores con beans gestionados (MBean)

Las API eXtreme Scale y los beans gestionados se han diseñado para permitir una fácil integración con las aplicaciones de supervisión de terceros. JConsole o MC4J son algunos ejemplos de consolas JMX (Java Management Extensions) ligeras que pueden utilizarse para analizar información sobre una topología de eXtreme Scale. También puede utilizar las API de programación para grabar implementaciones de adaptador en la instantánea o realizar un seguimiento de eXtreme Scale. WebSphere eXtreme Scale incluye una aplicación de supervisión de ejemplo que admite funciones de supervisión preconfiguradas que se pueden utilizar como plantilla para grabar programas de utilidad de supervisión personalizados más avanzados.

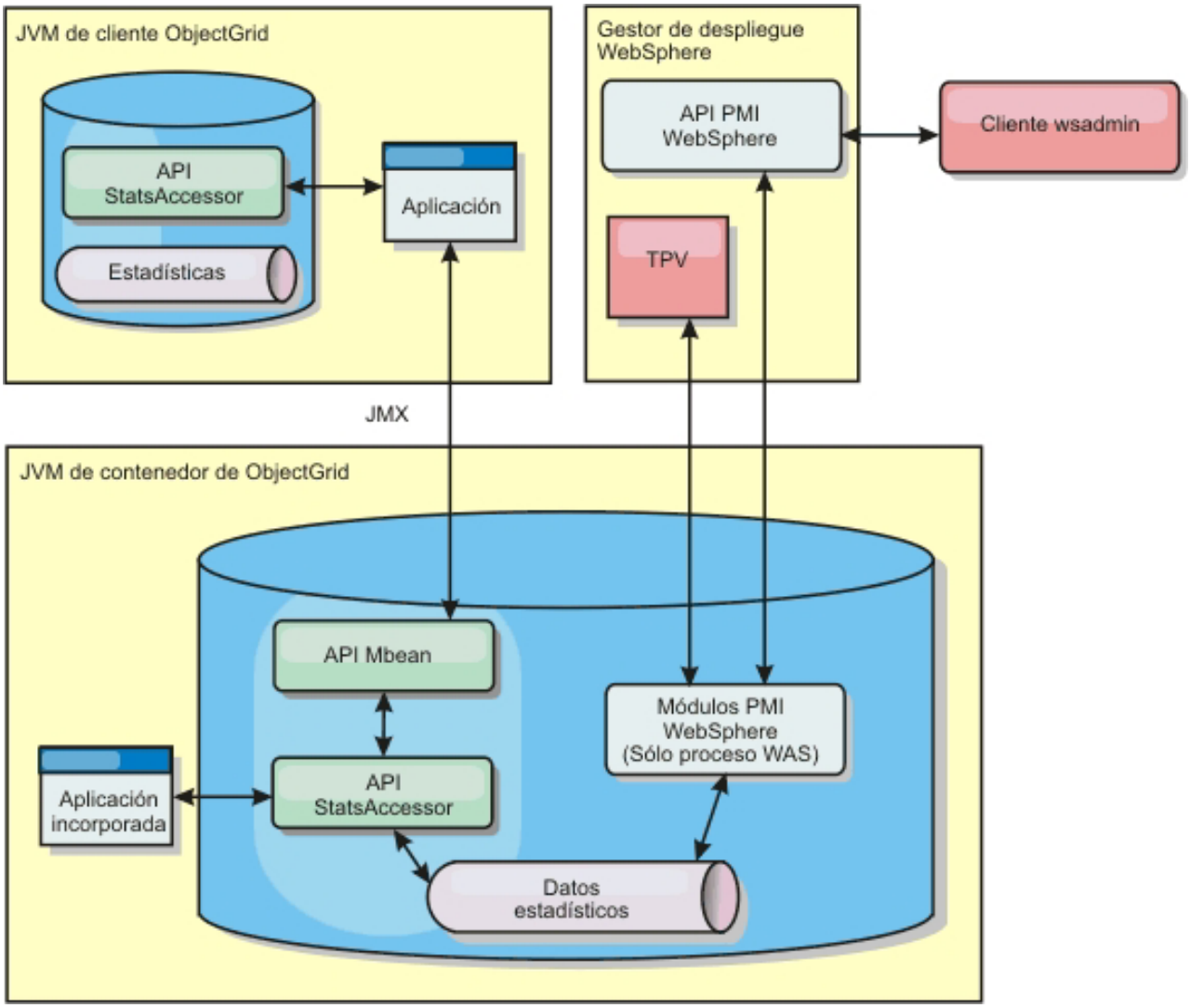


Figura 8. Visión general de MBean

Si desea más información, consulte "Utilización del programa de utilidad de ejemplo xsAdmin" en la página 287. Para obtener más información sobre la integración con aplicaciones de proveedores específicos, consulte los siguientes temas:

- Supervisión de eXtreme Scale con el agente IBM Tivoli Monitoring
- "Supervisión de eXtreme Scale con Hyperic HQ" en la página 301
- "Supervisión de aplicaciones de eXtreme Scale con CA Wily Introscope" en la página 297

Tareas relacionadas

Capítulo 8, “Supervisión del entorno de despliegue”, en la página 261
Puede utilizar API, MBeans, registros y programas de utilidad para supervisar el rendimiento del entorno de aplicación.

“Supervisión con la API de estadísticas” en la página 266
La API de estadísticas es la interfaz directa al árbol de estadísticas internas. De forma predeterminada las estadísticas están inhabilitadas, aunque pueden habilitarse estableciendo una interfaz StatsSpec. Una interfaz StatsSpec define cómo WebSphere eXtreme Scale debe supervisar estadísticas.

“Supervisión del rendimiento con PMI de WebSphere Application Server” en la página 270
WebSphere eXtreme Scale da soporte a PMI (Performance Monitoring Infrastructure) cuando se ejecuta en un servidor de aplicaciones WebSphere Application Server o WebSphere Extended Deployment. PMI recopila los datos de rendimiento de aplicaciones en tiempo de ejecución y proporciona interfaces que dan soporte a aplicaciones externas para supervisar datos de rendimiento. Puede utilizar la consola administrativa o la herramienta wsadmin para acceder a los datos de supervisión.

“Habilitación de PMI” en la página 271
Puede utilizar PMI (Performance Monitoring Infrastructure) de WebSphere Application Server para habilitar o inhabilitar estadísticas a cualquier nivel. Por ejemplo, puede elegir que se habiliten las estadísticas de proporción de coincidencias de correlación para una correlación determinada pero no así las estadísticas de número de entradas ni las estadísticas de tiempo de actualización por lotes del cargador. Puede habilitar PMI en la consola de administración o con scripts.

“Recuperar estadísticas de PMI” en la página 273
Al recuperar estadísticas de PMI, podrá ver el rendimiento de las aplicaciones eXtreme Scale.

“Utilización del programa de utilidad de ejemplo xsAdmin” en la página 287
Con el programa de utilidad de ejemplo xsAdmin puede formatear y mostrar información de texto sobre la topología de WebSphere eXtreme Scale. El programa de utilidad de ejemplo proporciona un método para analizar y descubrir los datos de despliegue actuales, y se puede utilizar como base para crear programas de utilidad personalizados.

“Supervisión de eXtreme Scale con Hyperic HQ” en la página 301
Hyperic HQ es una solución de supervisión de otro proveedor que está disponible de forma gratuita como una solución de código abierto o como un producto de empresa. WebSphere eXtreme Scale incluye un plug-in que permite a los agentes de Hyperic HQ descubrir los servidores de contenedor eXtreme Scale y crear informes y agregar estadísticas utilizando los beans de gestión de eXtreme Scale. Puede utilizar Hyperic HQ para supervisar los despliegues de eXtreme Scale autónomos.

“Supervisión con IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale” en la página 291
IBM Tivoli Enterprise Monitoring Agent es una solución de supervisión con muchas características que puede utilizar para supervisar bases de datos, sistemas operativos y servidores en entornos distribuidos y de host. WebSphere eXtreme Scale incluye un agente personalizado que puede utilizar para realizar introspecciones de los beans de gestión de eXtreme Scale. Esta solución funciona de forma eficaz tanto para los despliegues de eXtreme Scale autónomo, como para los despliegues de WebSphere Application Server.

Referencia relacionada

"Utilización de beans gestionados (MBeans) para administrar el entorno" en la página 286
Puede utilizar varios tipos distintos de MBeans JMX (Java Management Extensions) para administrar y supervisar despliegues. Cada MBean hace referencia a una entidad específica como, por ejemplo, una correlación, eXtreme Scale, servidor, grupo de réplicas o miembro del grupo de réplicas.

"Módulos PMI" en la página 275
Puede supervisar el rendimiento de las aplicaciones con los módulos PMI (Performance Monitoring Infrastructure).

"Módulos PMI" en la página 275
Puede supervisar el rendimiento de las aplicaciones con los módulos PMI (Performance Monitoring Infrastructure).

Supervisión con la API de estadísticas

La API de estadísticas es la interfaz directa al árbol de estadísticas internas. De forma predeterminada las estadísticas están inhabilitadas, aunque pueden habilitarse estableciendo una interfaz StatsSpec. Una interfaz StatsSpec define cómo WebSphere eXtreme Scale debe supervisar estadísticas.

Por qué y cuándo se efectúa esta tarea

Puede utilizar la API StatsAccessor local para consultar los datos y acceder a las estadísticas sobre cualquier instancia de ObjectGrid que está en la misma máquina virtual Java (JVM) que el código de ejecución. Si desea más información sobre las interfaces específicas, consulta la documentación de la API. Utilice los pasos siguientes para habilitar la supervisión del árbol de estadísticas internas.

1. Recupere el objeto StatsAccessor. La interfaz StatsAccessor sigue el patrón singleton. Por lo tanto, aparte de los problemas relacionados con el cargador de clases, debe existir una instancia de StatsAccessor para cada JVM . Esta clase hace las veces de interfaz principal para todas las operaciones de estadísticas locales. El siguiente código es un ejemplo sobre cómo recuperar la clase del descriptor de acceso. Llame a esta operación antes de cualquier otra llamada de ObjectGrid.

```
public class LocalClient {

   public static void main(String[] args) {

      // Recuperar un descriptor de contexto para StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

   }

}
```

2. Establezca la interfaz StatsSpec de cuadrícula. Establezca esta JVM para recopilar todas las estadísticas sólo en el nivel de ObjectGrid. Debe asegurarse de que una aplicación habilite todas las estadísticas que puedan ser necesarias antes de empezar las transacciones. En el siguiente ejemplo se establece la interfaz StatsSpec utilizando un campo contante estático y una serie de especificación. El uso de un campo constante estático es más fácil porque el campo ya ha definido la especificación. No obstante, si utiliza una serie de especificación, podrá habilitar cualquier combinación de estadísticas que sea necesaria.

```
public static void main(String[] args) {

      // Recuperar un descriptor de contexto para StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Establecer la especificación a través del campo estático
      StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);
      accessor.setStatsSpec(spec);
```

```
      // Establecer la especificación a través de la serie de especificación
      StatsSpec spec = new StatsSpec("og.all=enabled");
      accessor.setStatsSpec(spec);

}
```

3. Envíe las transacciones ala cuadrícula para obligar a que se recopilen los datos para la supervisión. Para recopilar datos prácticos para las estadísticas, debe enviar las transacciones a la cuadrícula. El siguiente extracto de código inserta un registro en MapA, que es un ObjectGridA. Dado que las estadísticas están en un nivel de ObjectGrid, cualquier correlación dentro de ObjectGrid genera los mismos resultados.

```
public static void main(String[] args) {

      // Recuperar un descriptor de contexto para StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Establecer la especificación a través del campo estático
      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");

      // Insertar unidad
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();
}
```

4. Consulte una StatsFact utilizando la API StatsAccessor. Cada vía de acceso de estadísticas está asociada a una interfaz StatsFact. La interfaz StatsFact es un marcador genérico que se utiliza para organizar y contener un objeto StatsModule. Para poder acceder al módulo de estadísticas real, se debe recuperar el objeto StatsFact.

```
public static void main(String[] args) {

      // Recuperar un descriptor de contexto para StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Establecer la especificación a través del campo estático
      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");

      // Insertar unidad
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();

      // Recuperar StatsFact

      StatsFact fact = accessor.getStatsFact(new String[] {"EmployeeGrid"},
    StatsModule.MODULE_TYPE_OBJECT_GRID);

}
```

5. Interactúe con el objeto StatsModule. El objeto StatsModule está incluido dentro de la interfaz StatsFact. Puede obtener una referencia al módulo utilizando la interfaz StatsFact. Como la interfaz StatsFact es una interfaz genérica, debe convertir el módulo devuelto en el tipo de StatsModule esperado. Puesto que esta tarea recopila estadísticas de eXtreme Scale, el objeto StatsModule devuelto se convierte en el tipo OGStatsModule. Una vez que se ha convertido el módulo, tendrá acceso a todas las estadísticas disponibles.

```
public static void main(String[] args) {

      // Recuperar un descriptor de contexto para StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Establecer la especificación a través del campo estático
      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");

      // Insertar unidad
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();

      // Recuperar StatsFact
      StatsFact fact = accessor.getStatsFact(new String[] {"EmployeeGrid"},
    StatsModule.MODULE_TYPE_OBJECT_GRID);

      // Recuperar módulo y hora
      OGStatsModule module = (OGStatsModule)fact.getStatsModule();
      ActiveTimeStatistic timeStat =
    module.getTransactionTime("Default", true);
      double time = timeStat.getMeanTime();

}
```

Conceptos relacionados

"Visión general de las estadísticas" en la página 261
Las estadísticas de WebSphere eXtreme Scale se calculan basándose en un árbol de estadísticas internas. La API StatsAccessor, los módulos PMI (Performance Monitoring Infrastructure) y la API MBean se generan desde el árbol interno.

"Herramientas del proveedor" en la página 290
WebSphere eXtreme Scale se puede supervisar utilizando varias soluciones populares de supervisión empresarial. Los agentes de plug-in se incluyen para IBM Tivoli Monitoring e Hyperic HQ, que supervisan WebSphere eXtreme Scale utilizando los beans de gestión a los que se puede acceder públicamente. CA Wily Introscope utiliza la instrumentación de métodos Java para capturar las estadísticas.

"Módulos de estadísticas"
WebSphere eXtreme Scale utiliza un modelo de estadísticas interno para rastrear y filtrar datos, que es la estructura subyacente que utilizan todas las vistas de datos para recopilar instantáneas de estadísticas. Puede utilizar varios métodos para recuperar la información de módulos de estadísticas.

Referencia relacionada

"Utilización de beans gestionados (MBeans) para administrar el entorno" en la página 286
Puede utilizar varios tipos distintos de MBeans JMX (Java Management Extensions) para administrar y supervisar despliegues. Cada MBean hace referencia a una entidad específica como, por ejemplo, una correlación, eXtreme Scale, servidor, grupo de réplicas o miembro del grupo de réplicas.

Módulos de estadísticas

WebSphere eXtreme Scale utiliza un modelo de estadísticas interno para rastrear y filtrar datos, que es la estructura subyacente que utilizan todas las vistas de datos para recopilar instantáneas de estadísticas. Puede utilizar varios métodos para recuperar la información de módulos de estadísticas.

Visión general

Las estadísticas en WebSphere eXtreme Scale se rastrean y contienen dentro de módulos StatsModules. Dentro del modelo de estadísticas, existen varios tipos de módulos de estadísticas:

OGStatsModule
Proporciona estadísticas para una instancia de ObjectGrid, incluidos tiempos de respuesta de transacciones.

MapStatsModule
Proporciona estadísticas para una sola correlación, incluido el número de entradas y la proporción de coincidencias.

QueryStatsModule
Proporciona estadísticas para consultas, incluido la creación del plan y los tiempos de ejecución.

AgentStatsModule
Proporciona estadísticas para los agentes de API de DataGrid, incluidos los tiempos de serialización y los tiempos de ejecución.

HashIndexStatsModule
Proporciona estadísticas para los tiempos de ejecución de mantenimiento y consulta de HashIndex.

SessionStatsModule
Proporciona estadísticas para el plug-in del gestor de sesiones HTTP.

Si desea detalles sobre los módulos de estadísticas, consulte la el paquete com.ibm.websphere.objectgrid.statsla documentación de la API.

Estadísticas en un entorno local

El modelo se organiza como un árbol n-ario (una estructura de árbol con el mismo grado para todos los nodos) compuesto por todos los tipos de StatsModule mencionados en la lista anterior. Debido a esta estructura de organización, cada nodo del árbol se representa mediante la interfaz StatsFact. La interfaz StatsFact puede representar un módulo individual o un grupo de módulos a efectos de agregación. Por ejemplo, si varios nodos finales en el árbol representan objetos MapStatsModule concretos, el nodo StatsFact padre relativo a estos nodos contiene estadísticas agregadas para todos los módulos hijos. Después de captar un objeto StatsFact, podrá utilizar la interfaz para recuperar el correspondiente StatsModule.

De forma muy parecida a una correlación de árbol, utilice una correspondiente vía de acceso o clave para recuperar un StatsFact específico. La vía de acceso es un valor String[] que consta de cada nodo que está junto a la vía de acceso al hecho solicitado. Por ejemplo, ha creado un ObjectGrid denominado ObjectGridA, que contiene dos correlaciones: MapA y MapB. La vía de acceso a StatsModule para MapA podría parecerse al siguiente `[ObjectGridA, MapA]`. La vía de acceso a las estadísticas agregadas para las dos correlaciones sería: `[ObjectGridA]`.

Estadísticas en un entorno distribuido

En un entorno distribuido, los módulos de estadísticas se recuperan utilizando una vía de acceso distinta. Dado que un servidor puede contener varias particiones, el árbol de estadísticas necesita realizar un seguimiento de la partición a la que pertenece cada módulo. Como resultado, la vía de acceso para consultar un objeto StatsFact concreto es distinto. Utilizando el ejemplo anterior, aunque añadiendo

que las correlaciones existen dentro de la partición 1, la vía de acceso es `[1, ObjectGridA, MapA]` para recuperar ese objeto StatsFact para MapA.

Tareas relacionadas

"Supervisión con la API de estadísticas" en la página 266
La API de estadísticas es la interfaz directa al árbol de estadísticas internas. De forma predeterminada las estadísticas están inhabilitadas, aunque pueden habilitarse estableciendo una interfaz StatsSpec. Una interfaz StatsSpec define cómo WebSphere eXtreme Scale debe supervisar estadísticas.

Supervisión del rendimiento con PMI de WebSphere Application Server

WebSphere eXtreme Scale da soporte a PMI (Performance Monitoring Infrastructure) cuando se ejecuta en un servidor de aplicaciones WebSphere Application Server o WebSphere Extended Deployment. PMI recopila los datos de rendimiento de aplicaciones en tiempo de ejecución y proporciona interfaces que dan soporte a aplicaciones externas para supervisar datos de rendimiento. Puede utilizar la consola administrativa o la herramienta wsadmin para acceder a los datos de supervisión.

Antes de empezar

- Puede utilizar PMI para supervisar el entorno cuando utiliza WebSphere eXtreme Scale junto con WebSphere Application Server.

Por qué y cuándo se efectúa esta tarea

WebSphere eXtreme Scale utiliza la característica PMI personalizada de WebSphere Application Server para añadir su propia instrumentación PMI. Con este enfoque, puede habilitar e inhabilitar WebSphere eXtreme Scale PMI con la consola administrativa o con las interfaces JMX (Java Management Extensions) de la herramienta wsadmin. Además, puede acceder a las estadísticas de WebSphere eXtreme Scale con las interfaces PMI y JMX estándares que utilizan las herramientas de supervisión, incluido Tivoli Performance Viewer.

1. Habilite eXtreme Scale PMI. Debe habilitar PMI para ver las estadísticas de PMI. Si desea más información, consulte "Habilitación de PMI" en la página 271.
2. Recupere las estadísticas de eXtreme Scale PMI. Vea el rendimiento de las aplicaciones deeXtreme Scale con Tivoli Performance Viewer. Si desea más información, consulte "Recuperar estadísticas de PMI" en la página 273.

Qué hacer a continuación

Para obtener más información sobre la herramienta wsadmin, consulte "Utilización del programa de utilidad wsadmin" en la página 285.

Conceptos relacionados

"Visión general de las estadísticas" en la página 261
Las estadísticas de WebSphere eXtreme Scale se calculan basándose en un árbol de estadísticas internas. La API StatsAccessor, los módulos PMI (Performance Monitoring Infrastructure) y la API MBean se generan desde el árbol interno.

"Herramientas del proveedor" en la página 290
WebSphere eXtreme Scale se puede supervisar utilizando varias soluciones populares de supervisión empresarial. Los agentes de plug-in se incluyen para IBM Tivoli Monitoring e Hyperic HQ, que supervisan WebSphere eXtreme Scale utilizando los beans de gestión a los que se puede acceder públicamente. CA Wily Introscope utiliza la instrumentación de métodos Java para capturar las estadísticas.

Referencia relacionada

"Utilización de beans gestionados (MBeans) para administrar el entorno" en la página 286
Puede utilizar varios tipos distintos de MBeans JMX (Java Management Extensions) para administrar y supervisar despliegues. Cada MBean hace referencia a una entidad específica como, por ejemplo, una correlación, eXtreme Scale, servidor, grupo de réplicas o miembro del grupo de réplicas.

"Módulos PMI" en la página 275
Puede supervisar el rendimiento de las aplicaciones con los módulos PMI (Performance Monitoring Infrastructure).

Habilitación de PMI

Puede utilizar PMI (Performance Monitoring Infrastructure) de WebSphere Application Server para habilitar o inhabilitar estadísticas a cualquier nivel. Por ejemplo, puede elegir que se habiliten las estadísticas de proporción de coincidencias de correlación para una correlación determinada pero no así las estadísticas de número de entradas ni las estadísticas de tiempo de actualización por lotes del cargador. Puede habilitar PMI en la consola de administración o con scripts.

Antes de empezar

El servidor de aplicaciones se debe haber iniciado y debe tener instalada una aplicación habilitada para eXtreme Scale. Para habilitar PMI con el uso de scripts, también debe poder iniciar la sesión y utilizar la herramienta wsadmin. Para obtener más información sobre la herramienta wsadmin, consulte el tema Herramienta wsadmin en el Information Center de WebSphere Application Server.

Por qué y cuándo se efectúa esta tarea

Utilice WebSphere Application Server PMI para proporcionar un mecanismo granular con el que poder habilitar o inhabilitar estadísticas a cualquier nivel. Por ejemplo, puede elegir que se habiliten las estadísticas de proporción de coincidencias de correlación para una correlación determinada pero no así las estadísticas de número de entradas ni las estadísticas de tiempo de actualización por lotes del cargador. Esta sección muestra cómo utilizar la consola administrativa y los scripts wsadmin para habilitar PMI de ObjectGrid.

- **Habilite PMI en la consola administrativa.**
  1. En la consola administrativa, pulse **Supervisión y ajuste → Performance Monitoring Infrastructure →** *nombre_servidor*.

2. Verifique que Performance Monitoring Infrastructure (PMI) se ha seleccionado. De forma predeterminada este valor está habilitado. Si el valor no está habilitado, seleccione el recuadro de selección y reinicie el servidor.
3. Pulse **Personalizado**. En el árbol de configuración, seleccione el módulo de correlaciones ObjectGrid y ObjectGrid. Habilite las estadísticas para cada módulo.

La categoría de tipo de transacción para estadísticas de ObjectGrid se crea en el tiempo de ejecución. Sólo puede ver las subcategorías de las estadísticas de ObjectGrid y de correlación en el separador **Tiempo de ejecución**.

- **Habilite PMI con el uso de scripts.**
  1. Abra un indicador de línea de mandatos. Vaya al directorio raíz_instalación/bin. Escriba wsadmin para iniciar la herramienta de línea de mandatos wsadmin.
  2. Modifique la configuración del tiempo de ejecución de PMI de eXtreme Scale. Verifique que PMI se ha habilitado para el servidor mediante los siguientes mandatos:

```
wsadmin>set s1 [$AdminConfig getid /Cell:CELL_NAME/Node:NODE_NAME/Server: APPLICATION_SERVER_NAME/]
wsadmin>set pmi [$AdminConfig list PMIService $s1]
wsadmin>$AdminConfig show $pmi.
```

     Si PMI no se ha habilitado, ejecute los mandatos siguientes para habilitar PMI:

```
wsadmin>$AdminConfig modify $pmi {{enable true}}
wsadmin>$AdminConfig save
```

     Si necesita habilitar PMI, reinicie el servidor.
  3. Establezca las variables para cambiar el conjunto de estadísticas por un conjunto personalizado utilizando los siguientes mandatos:

```
wsadmin>set perfName [$AdminControl completeObjectName type=Perf,
process=APPLICATION_SERVER_NAME,*]
wsadmin>set perfOName [$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. Establezca el conjunto de estadísticas que desea personalizar utilizando el siguiente mandato:

```
wsadmin>$AdminControl invoke_jmx $perfOName setStatisticSet $params $sigs
```

  5. Establezca las variables para habilitar las estadísticas de PMI de objectGridModule utilizando los siguientes mandatos:

```
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. Establezca la serie de estadísticas utilizando el siguiente mandato:

```
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. Establezca la serie de estadísticas utilizando el siguiente mandato:

```
wsadmin>$AdminControl invoke_jmx $perfOName setCustomSetString $params2 $sigs2
```

Estos pasos habilitan el PMI de tiempo de ejecución de eXtreme Scale, pero no modifican la configuración de PMI. Si reinicia el servidor de aplicaciones, los valores de PMI se pierden excepto para la habilitación de PMI principal.

Ejemplo

Puede efectuar los siguientes pasos para habilitar las estadísticas de PMI para la aplicación de ejemplo:

1. Inicie la aplicación utilizando la dirección web `http://host:puerto/ObjectGridSample`, donde el host y el puerto son el nombre del host y el número de puerto HTTP del servidor en el que se ha instalado el ejemplo.
2. En la aplicación de ejemplo, pulse ObjectGridCreationServlet, y luego pulse los botones de acción 1, 2, 3, 4 y 5 para generar acciones para ObjectGrid y correlaciones. No cierre esta página de servlet ahora.
3. En la consola administrativa, pulse **Supervisión y ajuste** → **Performance Monitoring Infrastructure** → *nombre_servidor* Pulse la pestaña **Tiempo de ejecución**.
4. Pulse el botón de selección **Personalizado**.
5. Expanda el módulo de correlaciones de ObjectGrid en el árbol de tiempo de ejecución y pulse el enlace clusterObjectGrid. Bajo el grupo de correlaciones de ObjectGrid, hay una instancia de ObjectGrid llamada clusterObjectGrid, y bajo el grupo clusterObjectGrid existen cuatro correlaciones: contadores, empleados, oficinas y sitios. En la instancia de ObjectGrids, existe la instancia de clusterObjectGrid y bajo dicha instancia hay un tipo de transacción llamado DEFAULT.
6. Puede habilitar las estadísticas que desee. Por ejemplo, puede habilitar una cantidad de entradas de correlación para la correlación de empleados y un tiempo de respuesta de transacción para el tipo de transacción DEFAULT.

Qué hacer a continuación

Una vez que se ha habilitado PMI, puede ver las estadísticas PMI con la consola administrativa o con el uso de scripts.

Conceptos relacionados

"Visión general de las estadísticas" en la página 261
Las estadísticas de WebSphere eXtreme Scale se calculan basándose en un árbol de estadísticas internas. La API StatsAccessor, los módulos PMI (Performance Monitoring Infrastructure) y la API MBean se generan desde el árbol interno.

Referencia relacionada

"Módulos PMI" en la página 275
Puede supervisar el rendimiento de las aplicaciones con los módulos PMI (Performance Monitoring Infrastructure).

Recuperar estadísticas de PMI

Al recuperar estadísticas de PMI, podrá ver el rendimiento de las aplicaciones eXtreme Scale.

Antes de empezar

- Habilite el rastreo de estadísticas de PMI para el entorno. Si desea más información, consulte "Habilitación de PMI" en la página 271.

- En las vías de acceso de esta tarea se da por supuesto que se recuperan estadísticas de la aplicación de ejemplo, aunque puede utilizar estas estadísticas para cualquier otra aplicación con pasos parecidos.
- Si utiliza la consola administrativa para recuperar estadísticas, debe poder iniciar la sesión en la consola administrativa. Si utiliza scripts, debe poder iniciar la sesión en wsadmin.

Por qué y cuándo se efectúa esta tarea

Puede recuperar estadísticas de PMI y verlas en Tivoli Performance Viewer efectuando los pasos en la consola administrativa o con scripts.

- Pasos en la consola administrativa
- Pasos en los scripts

Para obtener más información sobre las estadísticas que pueden recuperarse, consulte "Módulos PMI" en la página 275.

- Recupere estadísticas de PMI en la consola administrativa.
  1. En la consola administrativa, pulse **Supervisión y ajuste → Performance Viewer → Actividad actual**
  2. Seleccione el servidor que desee supervisar utilizando Tivoli Performance Viewer y luego habilite la supervisión.
  3. Pulse el servidor para ver la página de Performance Viewer.
  4. Expanda el árbol de configuración. Pulse **Correlaciones de ObjectGrid → clusterObjectGrid** seleccione **employees**. Expanda **ObjectGrids → clusterObjectGrid** y seleccione **DEFAULT**.
  5. En la aplicación de ejemplo de ObjectGrid, vaya al servlet ObjectGridCreationServlet, pulse el botón 1 y luego rellene las correlaciones. Puede ver las estadísticas en el visor.
- Recupere estadísticas de PMI con scripts.
  1. En el indicador de línea de mandatos, vaya al directorio `raíz_instalación/bin`. Escriba `wsadmin` para iniciar la herramienta wsadmin.
  2. Establezca las variables para el entorno utilizando los siguientes mandatos:

     ```
     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. Establezca las variables para obtener estadísticas de mapModule utilizando los siguientes mandatos:

     ```
     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. Obtenga estadísticas de mapModule utilizando el siguiente mandato:

     ```
     wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params $sigs
     ```

  5. Establezca las variables para obtener estadísticas de objectGridModule utilizando los siguientes mandatos:

     ```
     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. Obtenga las estadísticas de objectGridModule utilizando el siguiente mandato:

```
wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params2 $sigs2
```

Resultados

Puede ver las estadísticas en Tivoli Performance Viewer.

Conceptos relacionados

"Visión general de las estadísticas" en la página 261
Las estadísticas de WebSphere eXtreme Scale se calculan basándose en un árbol de estadísticas internas. La API StatsAccessor, los módulos PMI (Performance Monitoring Infrastructure) y la API MBean se generan desde el árbol interno.

Referencia relacionada

"Módulos PMI"
Puede supervisar el rendimiento de las aplicaciones con los módulos PMI (Performance Monitoring Infrastructure).

Módulos PMI

Puede supervisar el rendimiento de las aplicaciones con los módulos PMI (Performance Monitoring Infrastructure).

objectGridModule

objectGridModule contiene una estadística de tiempo: el tiempo de respuesta de la transacción. Una transacción se define como la duración entre la llamada del método Session.begin y la llamada del método Session.commit. Este intervalo de tiempo se considera el tiempo de respuesta de la transacción. El elemento raíz de objectGridModule, "root", hace las veces de punto de entrada de las estadísticas de WebSphere eXtreme Scale. Este elemento raíz tiene ObjectGrids como sus elementos hijo, que tienen tipos de transacción como elementos hijo. La estadística de tiempo de respuesta está asociado a cada tipo de transacción.

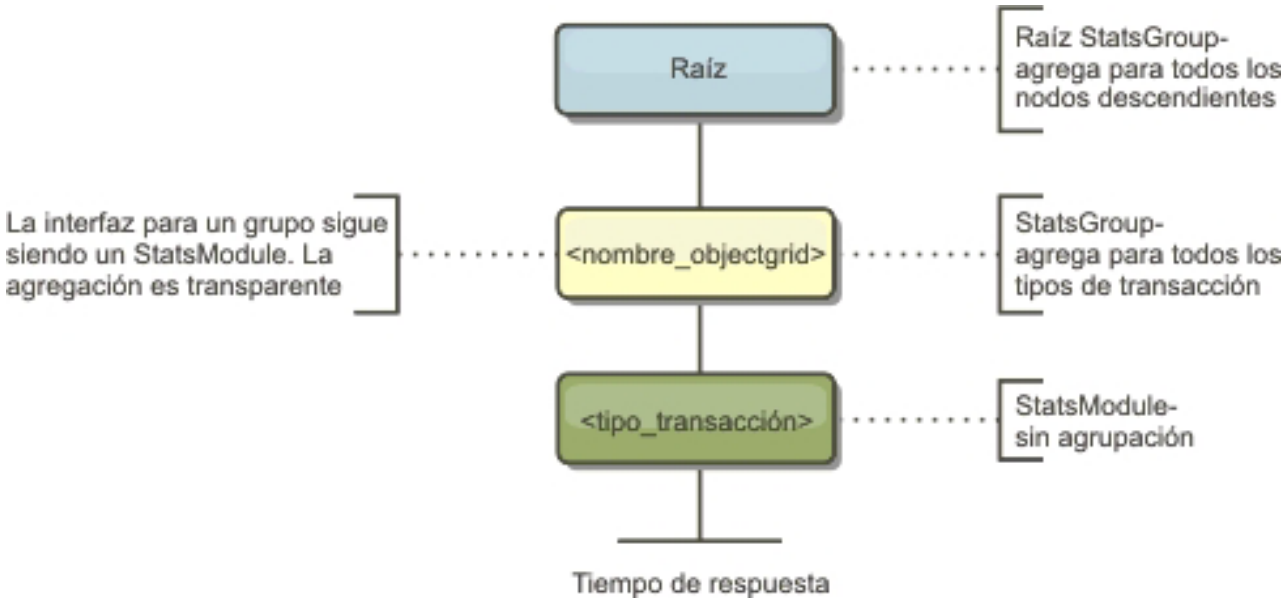


Figura 9. Estructura del módulo ObjectGridModule

El siguiente diagrama muestra un ejemplo de la estructura de ObjectGridModule. En este ejemplo, existen dos instancias de ObjectGrid en el sistema: el ObjectGrid A y el ObjectGrid B. La instancia A de ObjectGrid tiene dos tipos de transacciones: la A y la predeterminada. La instancia ObjectGrid B tiene sólo el tipo de transacción predeterminado.

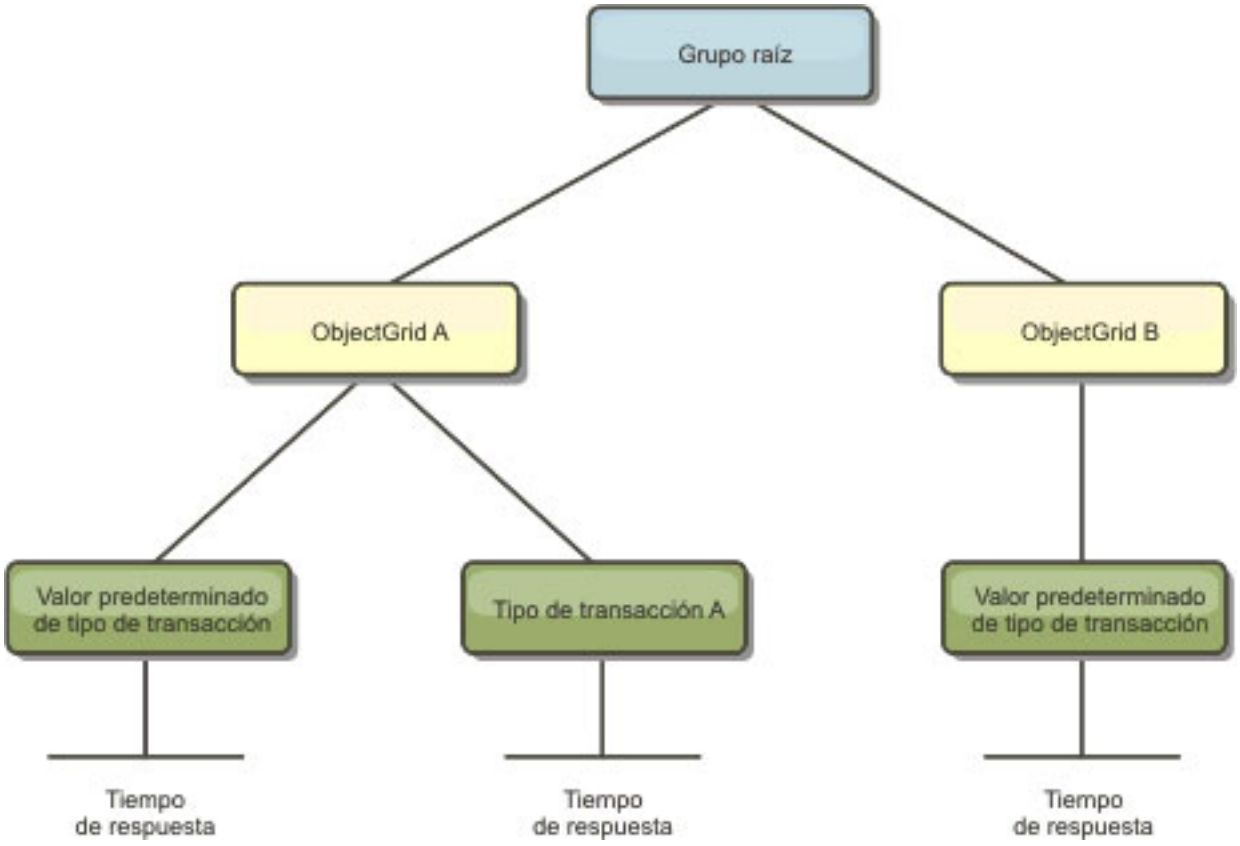


Figura 10. Ejemplo de estructura del módulo ObjectGridModule

Los tipos de transacción los definen los desarrolladores de transacciones porque conocen los tipos de transacciones que utilizan sus aplicaciones. El tipo de transacción se establece utilizando el siguiente método Session.setTransactionType(String):

```
/**
* Establece el tipo de transacción para futuras transacciones.
*
* Después de llamar a este método, todas las transacciones futuras tendrán el mismo
* tipo hasta que se establezca otro tipo de transacción. Si no se establece ningún
* tipo de transacción , se utiliza el tipo de transacción TRANSACTION_TYPE_DEFAULT
* predeterminado.
*
* Los tipos de transacción se usan principalmente para fines de seguimiento de datos
* estadísticos.
* Los usuarios pueden definir previamente los tipos de transacciones que se ejecutan en una
* aplicación.
La idea es clasificar las transacciones con las mismas características en una
* categoría (tipo), de modo que una estadística de tiempo de respuesta se pueda
* utilizar para realizar un seguimiento de cada tipo de transacción.
*
* Este seguimiento resulta útil cuando la aplicación tiene tipos diferentes de
* transacciones.
* Entre ellos, algunos tipos de transacciones, como las transacciones de
* actualización, tardan  más en procesarse que otras transacciones como, por
* ejemplo, las de sólo lectura. Utilizando  el tipo de transacción, se puede
* realizar un seguimiento de las transacciones diferentes por estadísticas
* diferentes, por lo que las estadísticas resultan más útiles.
*
* @param tranType el tipo de transacción para las transacciones futuras.
*/
void setTransactionType(String tranType);
```

El ejemplo siguiente establece el tipo de transacción en `updatePrice`:

```
// Establecer el tipo de transacción en updatePrice
// El período de tiempo entre session.begin() y session.commit() se reflejará
// en la estadística de tiempo "updatePrice".
```

```
session.setTransactionType("updatePrice");
session.begin();
map.update(stockId, new Integer(100));
session.commit();
```

La primera línea indica que el tipo de transacción subsiguiente es updatePrice. Existe una estadística updatePrice en la instancia ObjectGrid que se corresponde a la sesión del ejemplo. Utilizando las interfaces JMX (Java Management Extensions), puede obtener el tiempo de respuesta de la transacción para las transacciones updatePrice. También puede obtener la estadística agregada para todos los tipos de transacciones en la instancia ObjectGrid especificada.

mapModule

El mapModule contiene tres estadísticas que están relacionadas con las correlaciones de eXtreme Scale:

- **Proporción de coincidencias de la correlación** -*BoundedRangeStatistic*: efectúa un seguimiento de la proporción de coincidencias de una correlación. La proporción de coincidencias es un valor flotante entre 0 y 100 inclusive, que representa el porcentaje de coincidencias de una correlación en relación con las operaciones get de la correlación.
- **Número de entradas** - *CountStatistic*: efectúa un seguimiento del número de entradas de la correlación.
- **Tiempo de respuesta de la actualización por lotes del cargador**-*TimeStatistic*: efectúa un seguimiento del tiempo de respuesta que se utiliza para la operación de actualización por lotes del cargador.

El elemento raíz de mapModule, "root", hace las veces de punto de entrada para las estadísticas de la correlación de ObjectGrid. Este elemento raíz tiene ObjectGrids como elementos hijo, que tienen correlaciones como sus elementos hijo. Cada instancia de correlación tiene tres estadísticas listadas. En el diagrama siguiente se muestra la estructura de mapModule:

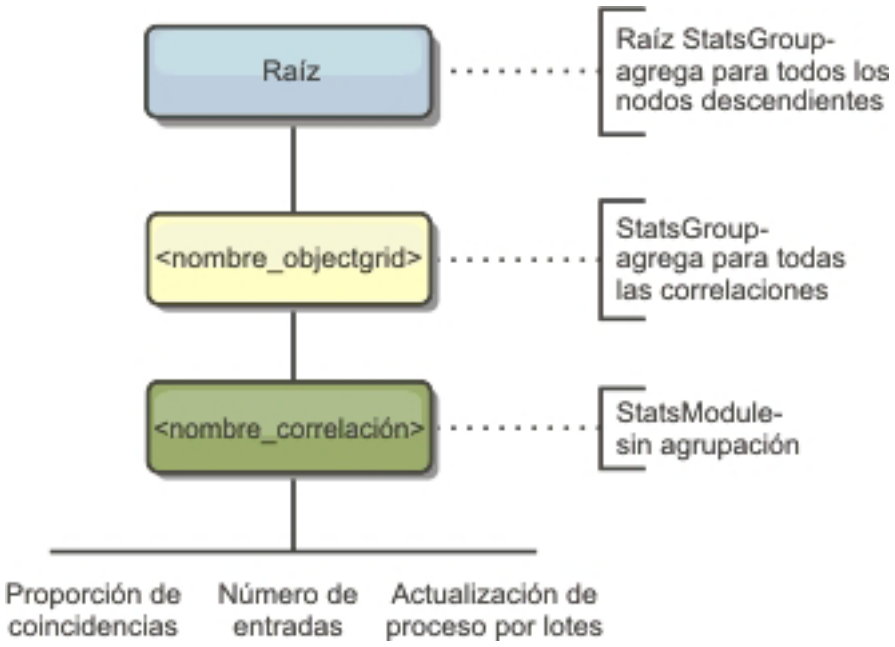
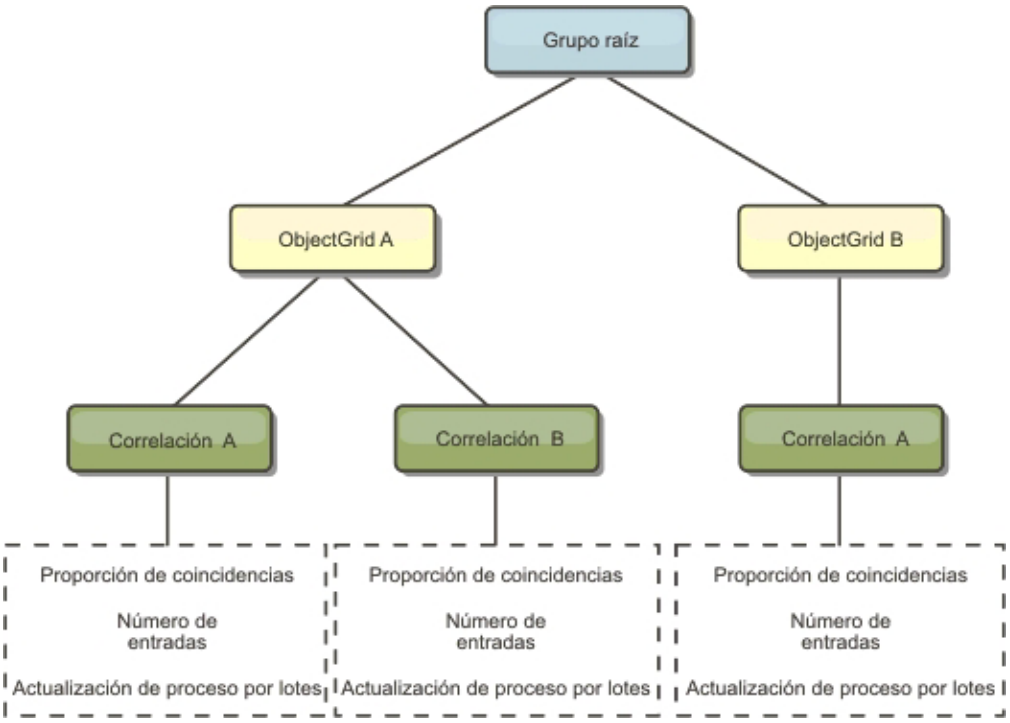


Figura 11. Estructura de mapModule

El siguiente diagrama muestra un ejemplo de la estructura de mapModule:

Figura 12. Ejemplo de la estructura del módulo mapModule



hashIndexModule

hashIndexModule contiene las siguientes estadísticas relacionadas con los índices de nivel de correlación:

- **Recuento de búsquedas** - *CountStatistic*: el número de invocaciones para la operación de búsqueda de índices.
- **Recuento de colisiones** - *CountStatistic*: el número de colisiones para la operación de búsqueda.
- **Recuento de anomalías** - *CountStatistic*: el número de anomalías para una operación de búsqueda.
- **Recuento de resultados** - *CountStatistic*: el número de claves devueltas de la operación de búsqueda.
- **Recuento de actualizaciones de proceso por lotes** - *CountStatistic*: el número de actualizaciones de proceso por lotes realizadas en relación con este índice. Cuando se cambia la correlación correspondiente en algún modo, el índice llamará a su método doBatchUpdate(). Esta estadística le indicará con que frecuencia se cambia o actualiza el índice.
- **Periodo de tiempo de la operación de búsqueda**-*TimeStatistic*: el intervalo de tiempo que la operación de búsqueda tarda en llevarse a cabo

El elemento raíz de hashIndexModule, "root", hace las veces de punto de entrada de las estadísticas de HashIndex. Este elemento raíz tiene ObjectGrids como elementos hijo, ObjectGrids tienen correlaciones como elementos hijo, que finalmente tienen HashIndexes como elementos hijo y nodos finales del árbol. Cada instancia de HashIndex tiene tres estadísticas listadas. En el diagrama

siguiente se muestra la estructura de hashIndexModule:

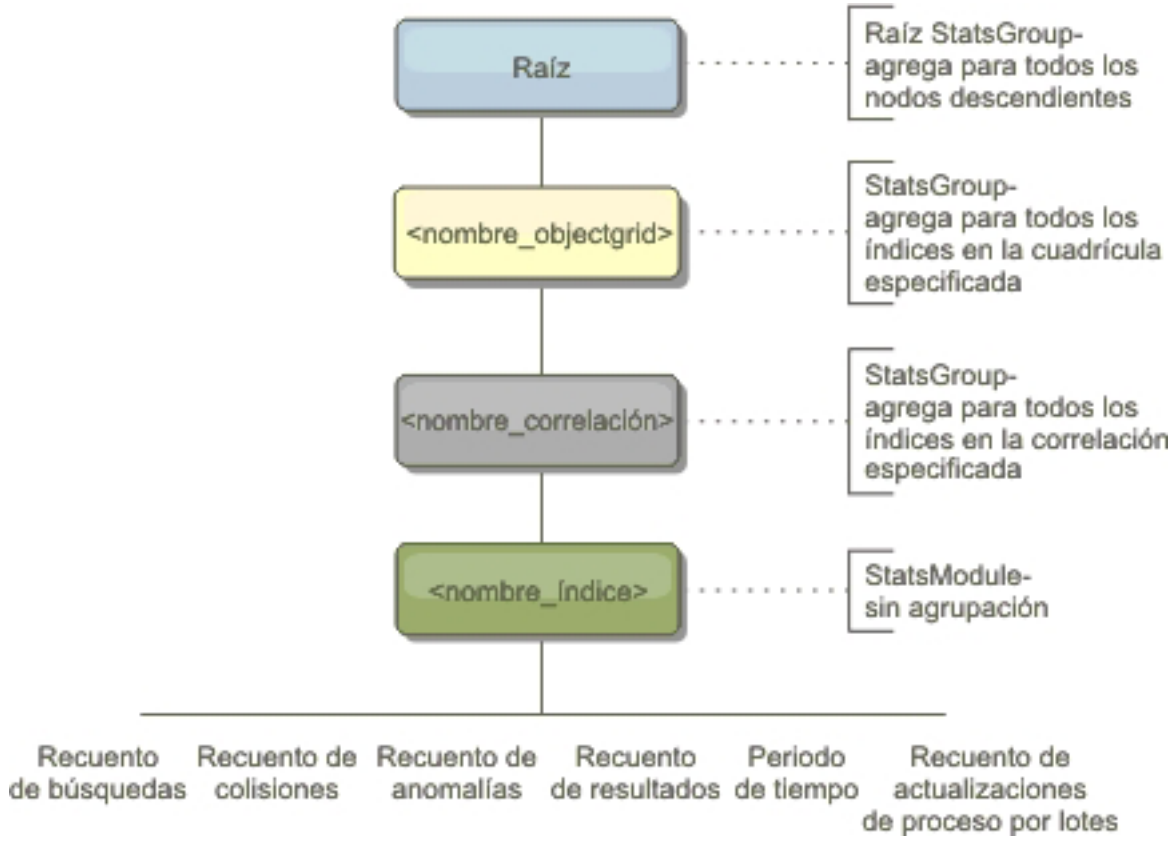


Figura 13. Estructura del módulo hashIndexModule

El siguiente diagrama muestra un ejemplo de la estructura de hashIndexModule:

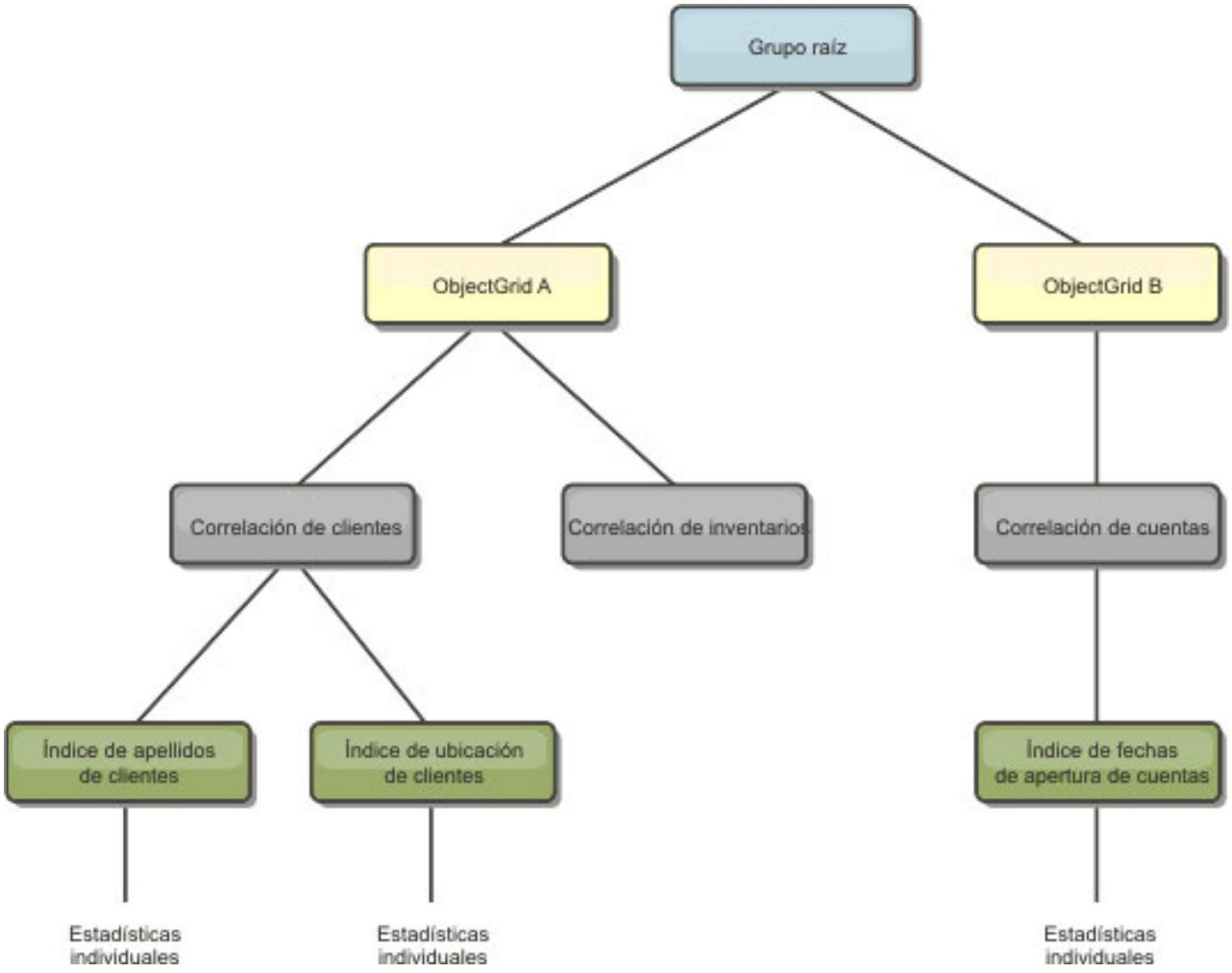


Figura 14. Ejemplo de estructura del módulo hashIndexModule

agentManagerModule

agentManagerModule contiene estadísticas relacionadas con los agentes de nivel de correlación:

- **Periodo de tiempo de reducción** - *TimeStatistic*: el intervalo de tiempo para que el agente termine la operación de reducción.
- **Periodo de tiempo total** - *TimeStatistic*: el intervalo de tiempo para que el agente complete todas las operaciones.
- **Periodo de tiempo de serialización de agente** - *TimeStatistic*: el intervalo de tiempo para serializar el agente.
- **Periodo de tiempo de inflación de agente** - *TimeStatistic*: el intervalo de tiempo que se tarda en inflar el agente en el servidor.
- **Periodo de tiempo de serialización de resultados** - *TimeStatistic*: el intervalo de tiempo para serializar los resultados de un agente.
- **Periodo de tiempo de inflación de resultados** - *TimeStatistic*: el intervalo de tiempo para inflar los resultados del agente.
- **Recuento de anomalías** - *CountStatistic*: el número de veces que el agente ha fallado.
- **Recuento de invocaciones** - *CountStatistic*: el número de veces que se ha invocado AgentManager.

- **Recuento de particiones** - *CountStatistic*: el número de particiones a las que se envía el agente.

El elemento raíz de agentManagerModule, "root", hace las veces de punto de entrada de las estadísticas de AgentManager. Este elemento raíz tiene ObjectGrids como elementos hijo, ObjectGrids tiene correlaciones como elementos hijo, que por último tienen instancias de AgentManager como elementos hijo y nodos finales del árbol. Cada instancia de AgentManager tiene las tres estadísticas listadas.

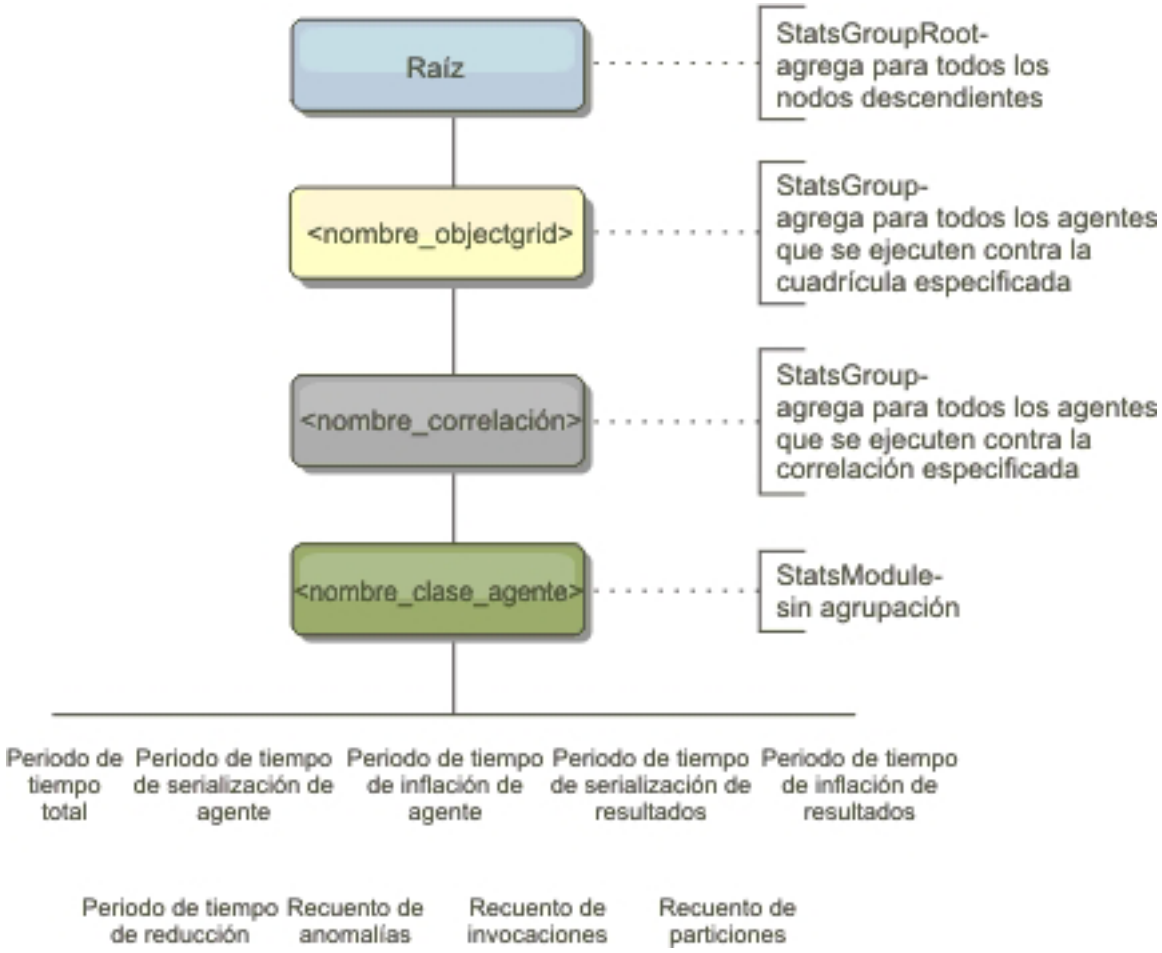


Figura 15. Estructura de agentManagerModule

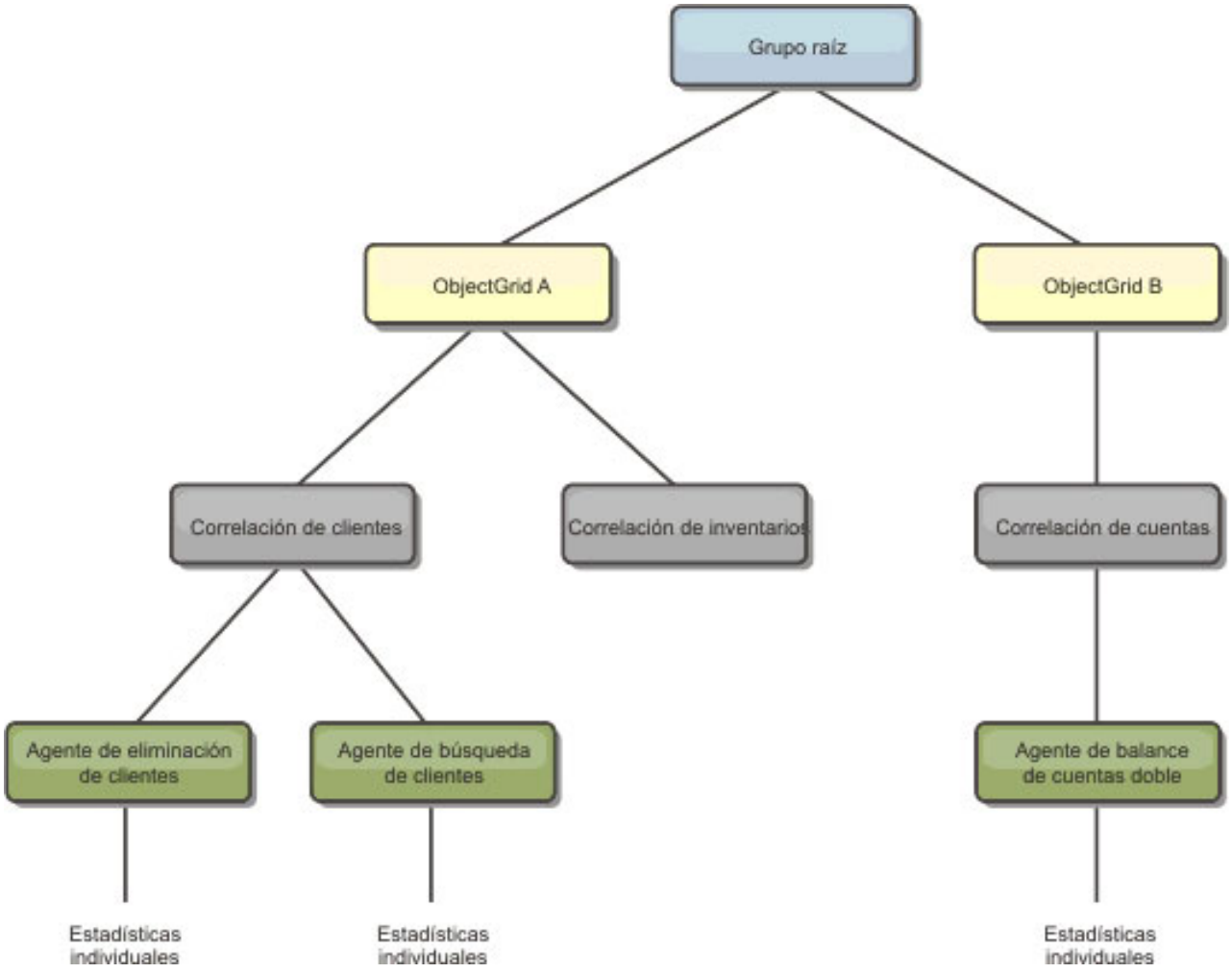


Figura 16. Ejemplo de la estructura de agentManagerModule

queryModule

queryModule contiene estadísticas relacionadas con las consultas de eXtreme Scale:

- **Tiempo de creación de plan** - *TimeStatistic*: el intervalo de tiempo para crear el plan de consulta.
- **Tiempo de ejecución** - *TimeStatistic*: el intervalo de tiempo para ejecutar la consulta.
- **Recuento de ejecuciones** - *CountStatistic*: el número de veces que se ha ejecutado la consulta.
- **Recuento de resultados** - *CountStatistic*: el recuento para cada conjunto de resultados de cada ejecución de consulta.
- **FailureCount** - *CountStatistic*: el número de veces que la consulta ha fallado.

El elemento raíz de queryModule, "root", hace las veces de punto de entrada de las estadísticas de Query. Este elemento raíz tiene ObjectGrids como elementos hijo, que tienen objetos Query como elementos hijo y nodos finales del árbol. Cada instancia de consulta tiene tres estadísticas listadas.

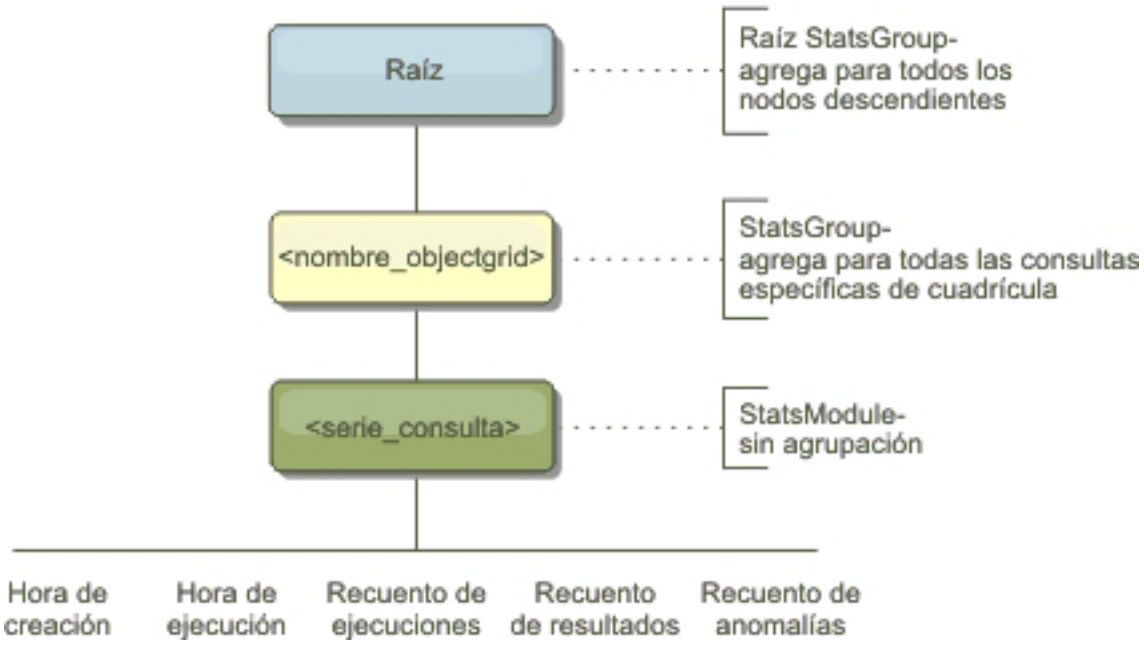


Figura 17. Estructura de queryModule

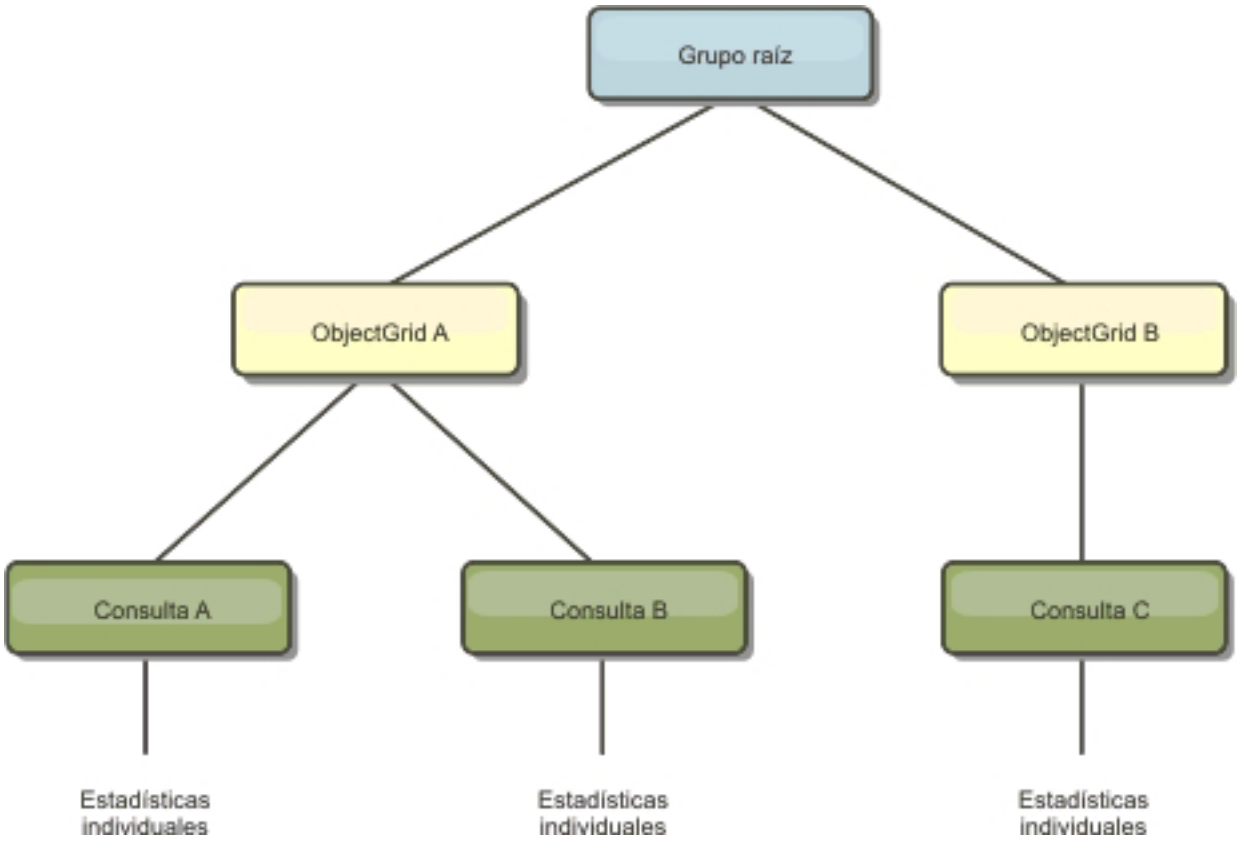


Figura 18. Ejemplo de la estructura de queryModule de QueryStats.jpg

Conceptos relacionados

"Visión general de las estadísticas" en la página 261
Las estadísticas de WebSphere eXtreme Scale se calculan basándose en un árbol de estadísticas internas. La API StatsAccessor, los módulos PMI (Performance Monitoring Infrastructure) y la API MBean se generan desde el árbol interno.

Tareas relacionadas

"Supervisión del rendimiento con PMI de WebSphere Application Server" en la página 270
WebSphere eXtreme Scale da soporte a PMI (Performance Monitoring Infrastructure) cuando se ejecuta en un servidor de aplicaciones WebSphere Application Server o WebSphere Extended Deployment. PMI recopila los datos de rendimiento de aplicaciones en tiempo de ejecución y proporciona interfaces que dan soporte a aplicaciones externas para supervisar datos de rendimiento. Puede utilizar la consola administrativa o la herramienta wsadmin para acceder a los datos de supervisión.

"Habilitación de PMI" en la página 271
Puede utilizar PMI (Performance Monitoring Infrastructure) de WebSphere Application Server para habilitar o inhabilitar estadísticas a cualquier nivel. Por ejemplo, puede elegir que se habiliten las estadísticas de proporción de coincidencias de correlación para una correlación determinada pero no así las estadísticas de número de entradas ni las estadísticas de tiempo de actualización por lotes del cargador. Puede habilitar PMI en la consola de administración o con scripts.

"Recuperar estadísticas de PMI" en la página 273
Al recuperar estadísticas de PMI, podrá ver el rendimiento de las aplicaciones eXtreme Scale.

"Utilización del programa de utilidad de ejemplo xsAdmin" en la página 287
Con el programa de utilidad de ejemplo xsAdmin puede formatear y mostrar información de texto sobre la topología de WebSphere eXtreme Scale. El programa de utilidad de ejemplo proporciona un método para analizar y descubrir los datos de despliegue actuales, y se puede utilizar como base para crear programas de utilidad personalizados.

Utilización del programa de utilidad wsadmin

Puede utilizar el programa de utilidad wsadmin proporcionado en WebSphere Application Server para acceder a la información de MBean.

Acceso a MBeans mediante la herramienta wsadmin

Ejecute la herramienta wsadmin desde el directorio bin de la instalación deWebSphere Application Server. En el siguiente ejemplo se recupera una vista de la colocación de fragmentos actual en un eXtreme Scale dinámico. Puede ejecutar wsadmin desde cualquier instalación donde se esté ejecutando eXtreme Scale. No tiene que ejecutar wsadmin en el servicio de catálogo.

```
$ 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>
```

Utilización de beans gestionados (MBeans) para administrar el entorno

Puede utilizar varios tipos distintos de MBeans JMX (Java Management Extensions) para administrar y supervisar despliegues. Cada MBean hace referencia a una entidad específica como, por ejemplo, una correlación, eXtreme Scale, servidor, grupo de réplicas o miembro del grupo de réplicas.

Las interfaces de MBean JMX yWebSphere eXtreme Scale

Cada MBean tiene métodos get que representan valores de atributos. Estos métodos get no se pueden invocar directamente desde el programa. La especificación JMX trata a los atributos de forma distinta de las operaciones. Puede ver atributos con una consola JMX de proveedor, así como realizar operaciones en el programa o con una consola JMX de proveedor.

Conceptos relacionados

"Visión general de las estadísticas" en la página 261
Las estadísticas de WebSphere eXtreme Scale se calculan basándose en un árbol de estadísticas internas. La API StatsAccessor, los módulos PMI (Performance Monitoring Infrastructure) y la API MBean se generan desde el árbol interno.

"Herramientas del proveedor" en la página 290
WebSphere eXtreme Scale se puede supervisar utilizando varias soluciones populares de supervisión empresarial. Los agentes de plug-in se incluyen para IBM Tivoli Monitoring e Hyperic HQ, que supervisan WebSphere eXtreme Scale utilizando los beans de gestión a los que se puede acceder públicamente. CA Wily Introscope utiliza la instrumentación de métodos Java para capturar las estadísticas.

Tareas relacionadas

Capítulo 8, "Supervisión del entorno de despliegue", en la página 261
Puede utilizar API, MBeans, registros y programas de utilidad para supervisar el rendimiento del entorno de aplicación.

"Supervisión con la API de estadísticas" en la página 266
La API de estadísticas es la interfaz directa al árbol de estadísticas internas. De forma predeterminada las estadísticas están inhabilitadas, aunque pueden habilitarse estableciendo una interfaz StatsSpec. Una interfaz StatsSpec define cómo WebSphere eXtreme Scale debe supervisar estadísticas.

"Supervisión del rendimiento con PMI de WebSphere Application Server" en la página 270
WebSphere eXtreme Scale da soporte a PMI (Performance Monitoring Infrastructure) cuando se ejecuta en un servidor de aplicaciones WebSphere Application Server o WebSphere Extended Deployment. PMI recopila los datos de rendimiento de aplicaciones en tiempo de ejecución y proporciona interfaces que dan soporte a aplicaciones externas para supervisar datos de rendimiento. Puede utilizar la consola administrativa o la herramienta wsadmin para acceder a los datos de supervisión.

Visión general del MBean gestionado

Puede utilizar los beans gestionados (MBeans) para rastrear las estadísticas en el entorno.

Antes de empezar

Para que se registren los atributos, debe habilitar las estadísticas. Puede habilitar las estadísticas de una de las formas siguientes:

- **Con el archivo de propiedades del servidor:**

  Puede habilitar las estadísticas en el archivo de propiedades del servidor con una entrada de clave-valor de `statsSpec=<StatsSpec>`. A continuación, algunos ejemplos de valores posibles:

  – Para habilitar todas las estadísticas, utilice `statsSpec=all=enabled` o `statisticsSpec=all=enabled`

  – Para habilitar sólo las estadísticas de ObjectGrid, utilice `statsSpec=og.all=enabled`. Para ver una descripción de todas las especificaciones estadísticas posibles, consulte la API StatsSpec en la documentación de la API.

  Si desea más información sobre el archivo de propiedades de servidor, consulte "Archivo de propiedades de servidor" en la página 197.

- **A través de programa:**

  También puede habilitar las estadísticas a través de programas con la interfaz StatsAccessor, que se recupera con la clase StatsAccessorFactory. Utilice esta interfaz en un entorno de cliente o cuando sea necesario supervisar un eXtreme Scale que se ejecute en el proceso actual.

Ejemplo

Si desea ver un ejemplo sobre cómo utilizar los beans gestionados, consulte "Utilización del programa de utilidad de ejemplo xsAdmin".

Utilización del programa de utilidad de ejemplo xsAdmin

Con el programa de utilidad de ejemplo xsAdmin puede formatear y mostrar información de texto sobre la topología de WebSphere eXtreme Scale. El programa de utilidad de ejemplo proporciona un método para analizar y descubrir los datos de despliegue actuales, y se puede utilizar como base para crear programas de utilidad personalizados.

Antes de empezar

Debe haber instalado WebSphere eXtreme Scale.

Por qué y cuándo se efectúa esta tarea

Puede utilizar el programa de utilidad de ejemplo xsAdmin para proporcionar comentarios sobre el diseño actual y el estado específico de la cuadrícula, como el contenido de la correlación. En este ejemplo, el diseño de la cuadrícula de esta tarea consta de una sola cuadrícula, denominada *ObjectGridA* con una correlación definida llamada *MapA*, que pertenece al conjunto de correlaciones, llamado *MapSetA*. Este ejemplo muestra cómo se puede mostrar todos los contenedores activos dentro de una cuadrícula e imprimir métricas filtradas relacionadas con el tamaño de correlación de *MapA*. Para ver todas las opciones de mandatos posibles, ejecute el programa de utilidad xsAdmin sin ningún argumento o con la opción **-help**.

1. En la línea de mandatos, establezca la variable de entorno JAVA_HOME.
   - UNIX `export JAVA_HOME=javaHome`

- Windows `set JAVA_HOME=javaHome`

2. Desplácese al directorio bin.
   ```
   cd objectGridRoot/bin
   ```
3. Inicie el programa de utilidad xsAdmin.
   - **Para mostrar la ayuda en línea, ejecute el siguiente mandato:** UNIX
     ```
     xsadmin.sh
     ```

     Windows
     ```
     xsadmin.bat
     ```

     Fíjese en la sección de argumentos necesarios del mensaje de ayuda, porque debe pasar sólo una de las opciones listadas para que el programa de utilidad funcione. Si no se especifica ninguna opción**-g** o **-m**, el programa de utilidad xsAdmin imprime información para cada cuadrícula de la topología.
   - **Para mostrar todos los contenedores en línea de una cuadrícula, ejecute el siguiente mandato:** UNIX
     ```
     xsadmin.sh -g ObjectGridA -m MapSetA -containers
     ```

     Windows
     ```
     xsadmin.bat -g ObjectGridA -m MapSetA -containers
     ```

     Se visualiza toda la información de contenedores. A continuación se muestra un ejemplo de la salida:
     ```
      Este programa de utilidad administrativo se proporciona únicamente como ejemplo
     y no se debe considerar como un componente totalmente soportado del producto
     WebSphere eXtreme Scale
     Conexión al servicio de catálogo en localhost:1099
     *** Mostrar todos los contenedores en línea para la cuadrícula - ObjectGridA
         & mapset - MapSetA
     Host: 192.168.0.186
       Container: server1_C-0, Server:server1, Zone:DefaultZone
         P:0 Primary
       Num containers matching = 1
       Total known containers = 1
       Total known hosts = 1
     ```
   - **Para mostrar el número de entradas de todas las correlaciones para una cuadrícula, ejecute el siguiente mandato:** UNIX
     ```
     xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA
     ```

     Windows
     ```
     xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA
     ```

     Se muestra el tamaño de la correlación especificada. A continuación se muestra un ejemplo de la salida:
     ```
     Este programa de utilidad administrativo se proporciona únicamente como ejemplo
     y no se debe considerar como un componente totalmente soportado del producto
     WebSphere eXtreme Scale
     Conexión al servicio de catálogo en localhost:1099

     ****Mostrando resultados para Grid - ObjectGridA, MapSet - MapSetA*****
     ```

```
*** Listado de Maps para server1 ***
 Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
Server Total: 0
```

- **Para especificar el puerto JMX para el servicio de catálogo, ejecute el siguiente mandato:** El programa de utilidad de ejemplo xsAdmin se conecta al servidor MBean que se ejecuta en un servidor de catálogo. Un servidor de catálogo puede ejecutarse en un proceso autónomo, el proceso WebSphere Application Server o incrustado dentro de un proceso de aplicaciones personalizado. Utilice la opción **-ch** para especificar el nombre de host del servicio de catálogo, y la opción **-p** para especificar el puerto de denominación del servicio de catálogo.

  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
  ```

  Se muestra el tamaño de la correlación especificada. A continuación se muestra un ejemplo de la salida:

  ```
  Este programa de utilidad administrativo se proporciona únicamente como ejemplo
  y no se debe considerar como un componente totalmente soportado del producto
  WebSphere eXtreme Scale
  Conexión al servicio de catálogo en CatalogMachine:6645

  *****Mostrando resultados para Grid - ObjectGridA, MapSet - MapSetA*****


  *** Listado de Maps para server1 ***
   Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
  Server Total: 0
  ```

- **Para conectarse a un servicio de catálogo que se aloja en un proceso de WebSphere Application Server, ejecute los siguientes pasos:**

  La opción **-dmgr** es necesaria al conectarse con un servicio de catálogo que alberga cualquier proceso o clúster de procesos de WebSphere Application Server. Utilice la opción **-ch** para especificar el nombre de host si no es `localhost`, y la opción **-p** para alterar temporalmente el puerto del programa de arranque del servicio de catálogo, que utilice el proceso BOOTSTRAP_ADDRESS. La opción **-p** sólo es necesaria si BOOTSTRAP_ADDRESS no está establecida en el valor predeterminado 9809.

  Nota: La versión autónoma de WebSphere eXtreme Scale no se puede utilizar para conectarse a un servicio de catálogo que se aloja en un proceso de WebSphere Application Server. Utilice el script xsAdmin incluido en el directorio *raíz_was*/bin, que está disponible cuando se instala WebSphere eXtreme Scale en WebSphere Application Server o WebSphere Application Server Network Deployment.

  a. Desplácese al directorio bin de WebSphere Application Server:

     ```
     cd wasRoot/bin
     ```

  b. 2. Inicie el programa de utilidad xsAdmin utilizando el siguiente mandato: UNIX

     ```
     xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
     ```

Windows

```
xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
```

c. Se muestra el tamaño de la correlación especificada.

```
Este programa de utilidad administrativo se proporciona únicamente como ejemplo
y no se debe considerar como un componente totalmente soportado del producto
WebSphere eXtreme Scale
Conexión al servicio de catálogo en localhost:9809

****Mostrando resultados para Grid - ObjectGridA, MapSet - MapSetA*****


*** Listado de Maps para server1 ***
 Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
Server Total: 0
```

Conceptos relacionados

"Visión general de las estadísticas" en la página 261
Las estadísticas de WebSphere eXtreme Scale se calculan basándose en un árbol de estadísticas internas. La API StatsAccessor, los módulos PMI (Performance Monitoring Infrastructure) y la API MBean se generan desde el árbol interno.

Referencia relacionada

"Módulos PMI" en la página 275
Puede supervisar el rendimiento de las aplicaciones con los módulos PMI (Performance Monitoring Infrastructure).

Herramientas del proveedor

WebSphere eXtreme Scale se puede supervisar utilizando varias soluciones populares de supervisión empresarial. Los agentes de plug-in se incluyen para IBM Tivoli Monitoring e Hyperic HQ, que supervisan WebSphere eXtreme Scale utilizando los beans de gestión a los que se puede acceder públicamente. CA Wily Introscope utiliza la instrumentación de métodos Java para capturar las estadísticas.

Tareas relacionadas

Capítulo 8, “Supervisión del entorno de despliegue”, en la página 261
Puede utilizar API, MBeans, registros y programas de utilidad para supervisar el rendimiento del entorno de aplicación.

“Supervisión con la API de estadísticas” en la página 266
La API de estadísticas es la interfaz directa al árbol de estadísticas internas. De forma predeterminada las estadísticas están inhabilitadas, aunque pueden habilitarse estableciendo una interfaz StatsSpec. Una interfaz StatsSpec define cómo WebSphere eXtreme Scale debe supervisar estadísticas.

“Supervisión del rendimiento con PMI de WebSphere Application Server” en la página 270
WebSphere eXtreme Scale da soporte a PMI (Performance Monitoring Infrastructure) cuando se ejecuta en un servidor de aplicaciones WebSphere Application Server o WebSphere Extended Deployment. PMI recopila los datos de rendimiento de aplicaciones en tiempo de ejecución y proporciona interfaces que dan soporte a aplicaciones externas para supervisar datos de rendimiento. Puede utilizar la consola administrativa o la herramienta wsadmin para acceder a los datos de supervisión.

“Supervisión de eXtreme Scale con Hyperic HQ” en la página 301
Hyperic HQ es una solución de supervisión de otro proveedor que está disponible de forma gratuita como una solución de código abierto o como un producto de empresa. WebSphere eXtreme Scale incluye un plug-in que permite a los agentes de Hyperic HQ descubrir los servidores de contenedor eXtreme Scale y crear informes y agregar estadísticas utilizando los beans de gestión de eXtreme Scale. Puede utilizar Hyperic HQ para supervisar los despliegues de eXtreme Scale autónomos.

“Supervisión con IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale”
IBM Tivoli Enterprise Monitoring Agent es una solución de supervisión con muchas características que puede utilizar para supervisar bases de datos, sistemas operativos y servidores en entornos distribuidos y de host. WebSphere eXtreme Scale incluye un agente personalizado que puede utilizar para realizar introspecciones de los beans de gestión de eXtreme Scale. Esta solución funciona de forma eficaz tanto para los despliegues de eXtreme Scale autónomo, como para los despliegues de WebSphere Application Server.

Referencia relacionada

“Utilización de beans gestionados (MBeans) para administrar el entorno” en la página 286
Puede utilizar varios tipos distintos de MBeans JMX (Java Management Extensions) para administrar y supervisar despliegues. Cada MBean hace referencia a una entidad específica como, por ejemplo, una correlación, eXtreme Scale, servidor, grupo de réplicas o miembro del grupo de réplicas.

Supervisión con IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale

IBM Tivoli Enterprise Monitoring Agent es una solución de supervisión con muchas características que puede utilizar para supervisar bases de datos, sistemas operativos y servidores en entornos distribuidos y de host. WebSphere eXtreme Scale incluye un agente personalizado que puede utilizar para realizar introspecciones de los beans de gestión de eXtreme Scale. Esta solución funciona de forma eficaz tanto para los despliegues de eXtreme Scale autónomo, como para los despliegues de WebSphere Application Server.

Antes de empezar

- Instale WebSphere eXtreme Scale versión 7.0.0 o posterior.
- Instale IBM Tivoli Monitoring versión 6.2.1 con el fixpack 2 o posterior.
- Instale el agente de sistema operativo Tivoli en cada servidor o sistema principal en el que se ejecuten los servidores eXtreme Scale.
- Instale el agente WebSphere eXtreme Scale, que puede descargar de forma gratuita desde el sitio IBM Open Process Automation Library (OPAL).

Complete los pasos siguientes para instalar y configurar Tivoli Monitoring Agent:

1. Instale Tivoli Monitoring Agent for WebSphere eXtreme Scale.

   Descargue la imagen de instalación de Tivoli y extraiga sus archivos en un directorio temporal.
2. Instale los archivos de soporte de aplicaciones de eXtreme Scale.

   Instale el soporte de aplicaciones de eXtreme Scale en cada uno de los siguientes despliegues.
   - Tivoli Enterprise Portal Server (TEPS)
   - Enterprise Desktop Client (TEPD)
   - Tivoli Enterprise Monitoring Server (TEMS)

   a. Desde el directorio temporal que ha creado, inicie una nueva ventana de mandato y ejecute el archivo ejecutable apropiado para la plataforma. El script de instalación detecta automáticamente el tipo de despliegue Tivoli (TEMS, TEPD o TEPS). Puede instalar cualquier tipo en un único host o en varios hosts; y los tres tipos de despliegue requieren la instalación de los archivos de soporte de aplicaciones del agente eXtreme Scale.

   b. En la ventana del **Instalador**, verifique que las selecciones de los componentes Tivoli desplegados son correctas. Pulse **Siguiente**.

   c. Si se le solicita, someta el nombre de host y las credenciales administrativas. Pulse **Siguiente**.

   d. Seleccione **Monitoring Agent for WebSphere eXtreme Scale**. Pulse **Siguiente**.

   e. Se le notificará qué acciones de instalación se llevarán a cabo. Pulse **Siguiente**, y podrá ver el progreso de la instalación hasta su finalización.

   Después de completar el procedimiento, se instalan todos los archivos de soporte de aplicaciones necesarios para el agente de WebSphere eXtreme Scale.
3. Instale el agente en cada uno de los nodos de eXtreme Scale.

   Instale un agente de sistema operativo Tivoli en cada uno de los sistemas. No tendrá que configurar ni iniciar este agente. Utilice la misma imagen de instalación del paso anterior para ejecutar el archivo ejecutable específico de la plataforma.

   Como directriz es necesario instalar sólo un agente por host. Cada agente es capaz de dar soporte a muchas instancias de servidores de eXtreme Scale. Para obtener un mejor rendimiento, utilice una instancia de agente para supervisar aproximadamente 50 servidores de eXtreme Scale.

   a. Desde la pantalla de bienvenida del asistente de instalación, pulse **Siguiente** para abrir la pantalla para especificar la información de la vía de acceso de instalación.

   b. Para el campo **Directorio de instalación de Tivoli Monitoring**, escriba o vaya a: `C:\IBM\ITM` (o `/opt/IBM/ITM`). Para el campo **Ubicación para soporte instalable**, verifique que el valor visualizado es correcto y pulse **Siguiente**.

c. Seleccione los componentes que desea añadir como, por ejemplo, **Realizar una instalación local de la solución** y pulse **Siguiente**.
d. Seleccione las aplicaciones para las que añade soporte seleccionando la aplicación como, por ejemplo, **Monitoring Agent for WebSphere eXtreme Scale**, y pulse **Siguiente**.
e. Podrá ver el progreso hasta que el soporte de aplicación se añada correctamente.

Nota: Repita estos pasos en cada uno de los nodos de eXtreme Scale. También puede utilizar una instalación silenciosa. Consulte el Centro de información de IBM Tivoli Monitoring si desea más información sobre la instalación silenciosa.

4. Configure el agente WebSphere eXtreme Scale.

   Es necesario configurar cada uno de los agentes instalados para supervisar cualquier servidor de catálogo, servidor de eXtreme Scale o ambos.

   Los pasos para configurar las plataformas Windows y UNIX son diferentes. La configuración de la plataforma Windows se completa con la interfaz de usuario **Gestionar servicios de supervisión Tivoli**. La configuración de las plataformas UNIX se basan en la línea de mandatos.

   Windows Utilice los pasos siguientes para configurar de forma inicial el agente en Windows Windows

   a. Desde la ventana **Manage Tivoli Enterprise Monitoring Services**, pulse **Inicio → Todos los programas → IBM Tivoli Monitoring → Manage Tivoli Monitoring Services**.
   b. Pulse con el botón derecho del ratón **Monitoring Agent for WebSphere eXtreme Scale** y seleccione **Configurar utilizando valores predeterminados**, que abre una ventana para crear una instancia exclusiva del agente.
   c. Elija un nombre exclusivo: por ejemplo, `instance1`, y pulse **Siguiente**.

   • Windows Si planifica supervisar los servidores eXtreme Scale autónomos, complete los pasos siguientes:

     a. Actualice los parámetros Java, asegúrese de que el valor de **Java Home** es correcto. Los argumentos de JVM se pueden dejar vacíos. Pulse **Siguiente**.
     b. Seleccione el tipo de **Tipo de conexión de servidor MBean**, Utilice `Servidor compatible con JSR-160` para los servidores eXtreme Scale autónomos. Pulse **Siguiente**.
     c. Si la seguridad está habilitada, actualice los valores **ID de usuario** y **Contraseña**. Deje el valor **URL de servicio JMX** tal cual. Altere temporalmente este valor más adelante. Deje el campo **Información de vía de acceso de clase JMX** tal cual. Pulse **Siguiente**.

     Para configurar los servidores para el agente en Windows, complete los pasos siguientes:

     a. Configure las instancias de subnodo de los servidores eXtreme Scale en el panel **Servidores de cuadrícula de WebSphere eXtreme Scale**. Si no existe ningún servidor de contenedor en el sistema, pulse **Siguiente** para seguir con el panel de servicio de catálogo.
     b. Si existen varios servidores de contenedor eXtreme Scale en el sistema, configure el agente para supervisar cada uno de los servidores.
     c. Puede añadir tantos servidores eXtreme Scale como necesite, si sus nombres y puertos son exclusivos, pulsando **Nuevo**. (Cuando se inicia un servidor eXtreme Scale, se debe especificar un valor JMXPort.)

d. Tras configurar los servidores de contenedor, pulse **Siguiente**, que le llevará al panel **Servidores de catálogos de WebSphere eXtreme Scale**.

e. Si no tiene ningún servidor de catálogo, pulse **Aceptar**. Si tiene servidores de catálogos, añada una nueva configuración para cada servidor, tal como ha hecho con los servidores de contenedor. De nuevo, elija un nombre exclusivo, preferentemente el mismo nombre que se utiliza cuando se inicia el servicio de catálogo. Pulse **Aceptar** para finalizar.

- Windows Si planifica supervisar los servidores para el agente en los servidores eXtreme Scale que se incorporan dentro de un proceso WebSphere Application Server, complete los pasos siguientes:

  a. Actualice los parámetros Java, asegúrese de que el valor de **Java Home** es correcto. Los argumentos de JVM se pueden dejar vacíos. Pulse **Siguiente**.

  b. Seleccione el **Tipo de conexión de servidor MBean**. Seleccione la versión de WebSphere Application Server que sea apropiada para el entorno. Pulse **Siguiente**.

  c. Asegúrese de que la información de WebSphere Application Server del panel es correcta. Pulse **Siguiente**.

  d. Añada sólo una definición de subnodo. Dé un nombre a la definición del subnodo, pero actualice la definición del puerto. Dentro del entorno WebSphere Application Server, el agente de nodo puede recopilar los datos de todos los servidores de aplicaciones gestionados por el agente de nodo que se ejecuta en el sistema. Pulse **Siguiente**.

  e. Si no existe ningún servidor de catálogo en el entorno, pulse **Aceptar**. Si tiene servidores de catálogos, añada una nueva configuración para cada servidor de catálogo, de forma similar a los servidores de contenedor. Elija un nombre exclusivo para el servicio de catálogos, preferentemente, el mismo nombre que utilice al iniciar el servicio de catálogos. Pulse **Aceptar** para finalizar.

Nota: No es necesario que los servidores de contenedor utilicen una ubicación compartida con el servicio de catálogos.

Ahora que el agente y los servidores están configurados y listos, en la ventana siguiente, pulse con el botón derecho del ratón `instance1` para iniciar el agente.

UNIX Para configurar el agente en la plataforma UNIX en la línea de mandatos, complete los pasos siguientes:

A continuación aparece un ejemplo para servidores autónomos que utiliza un tipo de conexión compatible con JSR160. El ejemplo muestra tres contenedores eXtreme Scale en el host único (rhea00b02) y las direcciones del receptor JMX son 15000,15001 y 15002 respectivamente. No hay ningún servidor de catálogo.

La salida del programa de utilidad de configuración aparece en *`cursiva monoespaciado`*, mientras que la respuesta del usuario aparece en **`negrita monoespaciado`**. (Si no era necesario ninguna respuesta del usuario, el valor predeterminado se seleccionó pulsando la tela Intro.) UNIX

```
rhea00b02 # ./itmcmd config -A xt
Se ha iniciado la configuración del agente...
Especifique el nombre de instancia (el valor predeterminado es: ):  inst1
¿Editar los valores de "Monitoring Agent for WebSphere eXtreme Scale"? [ 1=Sí, 2=No ] (el valor predeterminado es: 1):
¿Editar valores 'Java'? [ 1=Sí, 2=No ] (el valor predeterminado es: 1):
Directorio inicio de Java (el valor predeterminado es: C:\Archivos de programa\IBM\Java50):  /opt/OG61/java
Nivel de rastreo Java [ 1=Error, 2=Aviso, 3=Información, 4=Depuración mínima, 5=Depuración media, 6=Depuración máxima, 7=Todos ]
(el valor predeterminado es: 1):
Argumentos de JVM (el valor predeterminado es: ):
¿Editar valores de 'Conexión'? [ 1=Sí, 2=No ] (el valor predeterminado es: 1):
Tipo de conexión de servidor MBean [ 1=Servidor compatible con JSR-160, 2=WebSphere Application Server versión 6.0,
3=WebSphere Application Server versión 6.1, 4=WebSphere Application Server versión 7.0 ] (el valor predeterminado es: 1): 1
```

```
¿Editar valores de 'Servidor compatible con JSR-160'? [ 1=Sí, 2=No ] (el valor predeterminado es: 1):
ID de usuario JMX (el valor predeterminado es: ):
Especificar contraseña JMX (el valor predeterminado es: ):
Vuelva a especificar : contraseña JMX ( el valor predeterminado es: ):
URL de servicio de JMX (el valor predeterminado es: service:jmx:rmi:///jndi/rmi://localhost:port/objectgrid/MBeanServer):
----------------------------------------
Información de la classpath JMX
Vía de acceso base de JMX (el valor predeterminado es: ):
Vía de acceso de clases de JMX (el valor predeterminado es: ):
Directorios JAR de JMX (el valor predeterminado es: ):
¿Editar valores de 'Servicio de catálogo de WebSphere eXtreme Scale'? [ 1=Sí, 2=No ] (el valor predeterminado es: 1):  2
¿Editar valores de 'Servidores de cuadrícula de WebSphere eXtreme Scale'? [ 1=Sí, 2=No ] (el valor predeterminado es: 1): 1
¿No hay disponible ningún valor de 'Servidores de cuadrícula de WebSphere eXtreme Scale'?
Editar valores de 'Servidores de cuadrícula de WebSphere eXtreme Scale', [1=Añadir, 2=Editar, 3=Suprimir, 4=Siguiente, 5=Salir]
(el valor predeterminado es: 4): 1
Servidores de cuadrícula de WebSphere eXtreme Scale (el valor predeterminado es: ): rhea00b02_c0
URL del servicio JMX (el valor predeterminado es: service:jmx:rmi:///jndi/rmi://localhost:<puerto>/objectgrid/MBeanServer):
service:jmx:rmi:///jndi/rmi://localhost:15000/objectgrid/MBeanServer

Valores de 'Servidores de cuadrícula de WebSphere eXtreme Scale': WebSphere eXtreme Scale Grid Servers=ogx
Editar valores de 'Servidores de cuadrícula de WebSphere eXtreme Scale', [1=Añadir, 2=Editar, 3=Suprimir, 4=Siguiente, 5=Salir]
(el valor predeterminado es: 4): 1
Servidores de cuadrícula de WebSphere eXtreme Scale (el valor predeterminado es: ): rhea00b02_c1
URL del servicio JMX (el valor predeterminado es: service:jmx:rmi:///jndi/rmi://localhost:<puerto>/objectgrid/MBeanServer):
service:jmx:rmi:///jndi/rmi://localhost:15001/objectgrid/MBeanServer

Valores de 'Servidores de cuadrícula de WebSphere eXtreme Scale': WebSphere eXtreme Scale Grid Servers= rhea00b02_c1
Editar valores de 'Servidores de cuadrícula de WebSphere eXtreme Scale', [1=Añadir, 2=Editar, 3=Suprimir, 4=Siguiente, 5=Salir]
(el valor predeterminado es: 4): 1
Servidores de la cuadrícula de WebSphere eXtreme Scale (el valor predeterminado es: ): 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

Valores de 'Servidores de cuadrícula de WebSphere eXtreme Scale': WebSphere eXtreme Scale Grid Servers= rhea00b02_c2
Editar valores de 'Servidores de cuadrícula de WebSphere eXtreme Scale', [1=Añadir, 2=Editar, 3=Suprimir, 4=Siguiente, 5=Salir]
(el valor predeterminado es: 4): 5

¿Se conectará este agente a TEMS? [1=SÍ, 2=NO] (el valor predeterminado es: 1):
Nombre de host TEMS (el valor predeterminado es: rhea00b00):

Protocolo de red [ip, sna, ip.pipe o ip.spipe] (el valor predeterminado es: ip.pipe):

     Ahora seleccione el siguiente número de protocolo entre uno de los siguientes:
     - ip
     - sna
     - ip.spipe
     - 0 para ninguno
Network Protocol 2 (el valor predeterminado es: 0):
Número de puerto IP.PIPE (el valor predeterminado es: 1918):
Especifique el nombre de KDC_PARTITION (el valor predeterminado es: null):

¿Configurar la conexión para un TEMS secundario? [1=SÍ, 2=NO] (el valor predeterminado es: 2):
Especifique el Nombre de red principal opcional o 0 para "ninguno" (el valor predeterminado es: 0):
Se ha completado la configuración del agente...
```

El ejemplo anterior crea una instancia de agente denominada “inst1” y actualiza los valores de Java Home. Se configuran los servidores de contenedor eXtreme Scale, pero no se configura el servicio de catálogos.

Nota: El procedimiento anterior crea un archivo de texto con el siguiente formato en el directorio: `<instalación_ITM>/config/<host>_xt_<nombre instancia>.cfg`.

Ejemplo: `rhea00b02_xt_inst1.cfg`

Es mejor editar este archivo con el editor de texto sin formato que elija. A continuación, aparece un ejemplo del contenido de dicho archivo:

```
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 }  ]]
```

A continuación aparece un ejemplo que muestra una configuración en un despliegue de WebSphere Application Server:

```
rhea00b02 # ./itmcmd config -A xt
Se ha iniciado la configuración del agente...
Especifique el nombre de instancia (el valor predeterminado es: ):  inst1
```

```
¿Editar los valores de "Monitoring Agent for WebSphere eXtreme Scale"? [ 1=Sí, 2=No ] (el valor predeterminado es: 1): 1
¿Editar valores 'Java'? [ 1=Sí, 2=No ] (el valor predeterminado es: 1): 1
Inicio de Java (el valor predeterminado es: C:\Archivos de programa\IBM\Java50): /opt/WAS61/java
Nivel de rastreo Java [ 1=Error, 2=Aviso, 3=Información, 4=Depuración mínima, 5=Depuración media, 6=Depuración máxima, 7=Todo ]
(el valor predeterminado es: 1):
Argumentos de JVM (el valor predeterminado es: ):
¿Editar valores de 'Conexión'? [ 1=Sí, 2=No ] (el valor predeterminado es: 1):
Tipo de conexión de servidor MBean [ 1=Servidor compatible con JSR-160, 2=WebSphere Application Server versión 6.0,
3=WebSphere Application Server versión 6.1, 4=WebSphere Application Server versión 7.0 ] (el valor predeterminado es: 1): 4
¿Editar valores de 'WebSphere Application Server versión 7.0'? [ 1=Sí, 2=No ] (el valor predeterminado es: 1):ID de usuario WAS
(el valor predeterminado es: ):
Escribir la contraseña WAS (el valor predeterminado es: ):
Volver a escribir: contraseña WAS (el valor predeterminado es: ):
Nombre de host WAS (el valor predeterminado es: localhost): rhea00b02
Puerto WAS (el valor predeterminado es: 2809):
Protocolo de conector WAS [ 1=rmi, 2=soap ] (el valor predeterminado es: 1):
Nombre de perfil WAS (el valor predeterminado es: ): default
---------------------------------------
Información de classpath WAS
Vías de acceso básicas de WAS (el valor predeterminado es: C:\Archivos de programa\IBM\WebSphere\AppServer;/opt/IBM/WebSphere/AppServer):
/opt/WAS61
Classpath WAS (el valor predeterminado es: runtimes/com.ibm.ws.admin.client_6.1.0.jar;runtimes/com.ibm.ws.ejb.thinclient_7.0.0.jar):
Directorios JAR de WAS (el valor predeterminado es: lib;plugins):
¿Editar valores de 'Servidores de cuadrícula de WebSphere eXtreme Scale'? [ 1=Sí, 2=No ] (el valor predeterminado es: 1):
¿No hay ningún valor disponible de 'Servidores de cuadrícula de WebSphere eXtreme Scale'?
Editar valores de 'Servidores de cuadrícula de WebSphere eXtreme Scale', [1=Añadir, 2=Editar, 3=Suprimir, 4=Siguiente, 5=Salir]
(el valor predeterminado es: 4): 1
Servidores de cuadrícula de WebSphere eXtreme Scale (el valor predeterminado es: ): rhea00b02
URL de servicio JMX (el valor predeterminado es: service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer):

Valores de 'Servidores de cuadrícula de WebSphere eXtreme Scale': Servidores de cuadrícula de WebSphere eXtreme Scale=rhea00b02
Editar valores de 'Servidores de cuadrícula de WebSphere eXtreme Scale', [1=Añadir, 2=Editar, 3=Suprimir, 4=Siguiente, 5=Salir]
(el valor predeterminado es: 4): 5
¿Editar valores de 'Servicio de catálogo de WebSphere eXtreme Scale'? [ 1=Sí, 2=No ] (el valor predeterminado es: 1):  2
¿Este agente se conectará a un TEMS? [1=SÍ, 2=NO] (el valor predeterminado es: 1):
Nombre de host de TEMS (el valor predeterminado es: rhea00b02):

Protocolo de red [ip, sna, ip.pipe o ip.spipe] (el valor predeterminado es: ip.pipe):

     Ahora seleccione el siguiente número de protocolo entre uno de los siguientes:
     - ip
     - sna
     - ip.spipe
     - 0 para ninguno
Network Protocol 2 (el valor predeterminado es: 0):
Número de puerto IP.PIPE (el valor predeterminado es: 1918):
Especifique el nombre de KDC_PARTITION (el valor predeterminado es: null):

¿Configurar la conexión para un TEMS secundario? [1=SÍ, 2=NO] (el valor predeterminado es: 2):
Especifique el Nombre de red principal opcional o 0 para "ninguno" (el valor predeterminado es: 0):
Ha finalizado la configuración del agente...
rhea00b02 #
```

Para los despliegues de WebSphere Application Server, no es necesario que cree varios subnodos. El agente eXtreme Scale se conecta al agente de nodo para recopilar toda la información de los servidores de aplicaciones de los que es responsable.

`SECTION=CAT` indica una línea de servicio de catálogo mientras que `SECTION=OGS` indica una línea de configuración de servidor de eXtreme Scale.

5. Configure los servidores eXtreme Scale.

   Cuando se inician los servidores contenedor eXtreme Scale, sin especificar el argumento **-JMXServicePort**, se asigna un servidor MBean a un puerto dinámico. El agente necesita saber con anticipación con qué puerto se va a comunicar. El agente no funciona con puertos dinámicos.

   Cuando inicie los servidores, deberá especificar el argumento `-JMXServicePort <número_puerto>` cuando inicie el servidor eXtreme Scale utilizando el mandato `startOgServer.sh | .bat`. Ejecutar este mandato garantiza que el servidor del proceso está a la escucha de un puerto estático definido previamente.

   Para ver los ejemplos anteriores en una instalación UNIX, se deben iniciar dos servidores eXtreme Scale con los puertos establecidos:

   a. "-JMXServicePort" "15000" (para rhea00b02_c0)

   b. "-JMXServicePort" "15001" (para rhea00b02_c1)

   a. Inicie el agente WebSphere eXtreme Scale.

Dando por supuesto que se ha creado la instancia `inst1`, como en el ejemplo anterior, emita los siguientes mandatos.

1) `cd <instalación_ITM>/bin`
2) `itmcmd agent –o inst1 start xt`

b. Detenga el agente WebSphere eXtreme Scale.

Dando por supuesto que se ha creado la instancia “inst1”, como en el ejemplo anterior, emita los siguientes mandatos.

1) `cd <instalación_ITM>/bin`
2) `itmcmd agent –o inst1 stop xt`

Resultados

Después de configurar e iniciar todos los servidores, los datos de MBeans se visualizan en la consola de IBM Tivoli Portal. Los espacios de trabajo definidos previamente muestran gráficos y métricas de datos en cada nivel de nodo.

Están definidos los siguientes espacios de trabajo: el nodo de **Servidores de cuadrícula de WebSphere eXtreme Scale** para todos los nodos supervisados.

- Vista de transacciones de eXtreme Scale
- Vista de fragmento primario de eXtreme Scale
- Vista de memoria de eXtreme Scale
- Vista de ObjectMap de eXtreme Scale

También puede configurar sus propios espacios de trabajo. Si desea más información, consulte la información sobre cómo personalizar los espacios de trabajo en el centro de información de IBM Tivoli Monitoring.

Conceptos relacionados

“Herramientas del proveedor” en la página 290
WebSphere eXtreme Scale se puede supervisar utilizando varias soluciones populares de supervisión empresarial. Los agentes de plug-in se incluyen para IBM Tivoli Monitoring e Hyperic HQ, que supervisan WebSphere eXtreme Scale utilizando los beans de gestión a los que se puede acceder públicamente. CA Wily Introscope utiliza la instrumentación de métodos Java para capturar las estadísticas.

“Visión general de las estadísticas” en la página 261
Las estadísticas de WebSphere eXtreme Scale se calculan basándose en un árbol de estadísticas internas. La API StatsAccessor, los módulos PMI (Performance Monitoring Infrastructure) y la API MBean se generan desde el árbol interno.

Supervisión de aplicaciones de eXtreme Scale con CA Wily Introscope

CA Wily Introscope es un producto de gestión de otro proveedor que puede utilizar para detectar y diagnosticar problemas de rendimiento en entornos de aplicaciones de empresa. eXtreme Scale incluye detalles sobre cómo configurar CA Wily Introscope para realizar introspecciones en las partes de selección del tiempo de ejecución de eXtreme Scale para ver y validar rápidamente las aplicaciones eXtreme Scale. CA Wily Introscope funciona de forma eficaz tanto para los despliegues autónomos, como para los despliegues de WebSphere Application Server.

Visión general

Para supervisar aplicaciones de eXtreme Scale con CA Wily Introscope, debe poner valores en los archivos ProbeBuilderDirective (PBD) que le proporcionan acceso a la información de supervisión para eXtreme Scale.

Atención: Los puntos de instrumentación para Introscope podrían cambiar con cada fixpack o release. Al instalar un nuevo fixpack o release, consulte la documentación para ver cualquier cambio en los puntos de instrumentación.

Puede configurar los archivos CA Wily Introscope ProbeBuilderDirective (PBD) para supervisar las aplicaciones de eXtreme Scale. CA Wily Introscope es un producto de gestión de aplicaciones con el que puede detectar, desencadenar y diagnosticar de forma proactiva los problemas en los entornos complejos, compuestos y de aplicación web.

Valores de archivos PBD para supervisar el servicio de catálogo

Puede utilizar uno o más de los siguientes valores en el archivo PBD para supervisar el servicio de catálogo.

```
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}"
```

Clases para supervisar el servicio de catálogo

HAControllerImpl

La clase HAControllerImpl maneja sucesos de comentarios y ciclo de vida del grupo principal. Puede supervisar esta clase para obtener una indicación de los cambios y estructura del grupo principal.

ServerAgent

La clase ServerAgent se ocupa de comunicar sucesos de grupos principales con el servicio de catálogo. Puede supervisar las diversas llamadas de pulsaciones para encontrar sucesos importantes.

PlacementServiceImpl

La clase PlacementServiceImpl coordina los contenedores. Puede utilizar los métodos en esta clase para supervisar sucesos de colocación y unión de servidores.

BalanceGridEventListener

La clase BalanceGridEventListener controla el liderazgo del catálogo. Puede supervisar esta clase para obtener una indicación de qué servicio de catálogo actúa actualmente como líder.

Valores de archivos PBD para supervisar los contenedores

Puede utilizar uno o más de los siguientes valores del archivo PBD para supervisar los contenedores.

```
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}"
```

Clases para supervisar los contenedores

ShardImpl

La clase ShardImpl tiene el método processMessage. El método processMessage es el método para las solicitudes de cliente. Con este método, puede obtener los recuentos de solicitudes y tiempos de respuesta del lado del servidor. Observando los recuentos a lo largo de todos los servidores y supervisando la utilización del almacenamiento dinámico, puede determinar si la cuadrícula está equilibrada.

CheckpointIterator

La clase CheckpointIterator tiene la llamada al método activateListener que coloca los primarios en modalidad de igual. Cuando los primarios se colocan en modalidad de igual, al réplica está actualizada con el primario una vez el método finaliza. Cuando una réplica se regenera a partir de un primario completo, esta operación puede tardar bastante tiempo. El sistema no se recupera totalmente hasta que la operación finaliza, de modo que puede utilizar esta clase para supervisar el progreso de la operación.

CommittedLogSequenceListenerProxy

La clase CommittedLogSequenceListenerProxy tiene dos métodos de interés. El método applyCommitted se ejecuta para cada transacción y sendApplyCommitted se ejecuta cuando la réplica extrae información. La proporción de la frecuencia en que estos dos métodos se ejecutan puede darle algún indicio de hasta qué punto la réplica puede mantener el ritmo del primario.

Valores de archivos PBD para supervisar los clientes

Puede utilizar uno o más de los siguientes valores en el archivo PBD para supervisar los clientes.

```
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}"
```

Clases para supervisar los clientes

ORBClientCoreMessageHandler

La clase ORBClientCoreMessageHandler es responsable de enviar solicitudes de aplicación a los contenedores. Puede supervisar el método sendMessage para el tiempo de respuesta del cliente y el número de solicitudes.

ClusterStore

La clase ClusterStore mantiene la información de direccionamiento en el lado del cliente.

BaseMap

La clase BaseMap tiene el método evictMapEntries que se invoca cuando el desalojador desea eliminar entradas de la correlación.

SelectionServiceImpl

La clase SelectionServiceImpl toma las decisiones de direccionamiento. Si el

cliente toma decisiones relacionadas con la migración tras error, puede utilizar esta clase para ver las acciones que se han completado de las decisiones.

ObjectGridImpl
La clase ObjectGridImpl tiene el método getSession que puede supervisar para ver el número de solicitudes para este método.

Supervisión de eXtreme Scale con Hyperic HQ

Hyperic HQ es una solución de supervisión de otro proveedor que está disponible de forma gratuita como una solución de código abierto o como un producto de empresa. WebSphere eXtreme Scale incluye un plug-in que permite a los agentes de Hyperic HQ descubrir los servidores de contenedor eXtreme Scale y crear informes y agregar estadísticas utilizando los beans de gestión de eXtreme Scale. Puede utilizar Hyperic HQ para supervisar los despliegues de eXtreme Scale autónomos.

Antes de empezar

- Este conjunto de instrucciones es para Hyperic versión 4.0. Si tiene una versión más reciente de Hyperic, consulte la documentación de Hyperic si desea más información como, por ejemplo, los nombres de vía de acceso y cómo iniciar agentes y servidores.
- Descargue las instalaciones de agente y servidor de Hyperic. Debe estar en ejecución una instalación de servidor. Para detectar todos los servidores eXtreme Scale, un agente Hyperic debe estar en ejecución en cada máquina en la que se ejecuta un servidor eXtreme Scale. Consulte el sitio web de Hyperic si desea información de descarga y soporte de documentación.
- Debe tener acceso a los archivos `objectgrid-plugin.xml` `yhqplugin.jar`. Estos archivos se encuentran en el directorio `objectgridRoot/hyperic/etc`.

Por qué y cuándo se efectúa esta tarea

Mediante la integración de eXtreme Scale con el software de supervisión Hyperic HQ, puede supervisar y visualizar mediante gráficos las métricas sobre el rendimiento del entorno. Configure esta integración utilizando una implementación de plug-in en cada agente.

1. Inicie los servidores eXtreme Scale. El plug-in Hyperic consulta los procesos locales para conectarse al Máquinas virtuales Java que ejecuta eXtreme Scale. Para conectarse correctamente a Máquinas virtuales Java , todos los servidores deben estar iniciados con la opción **-jmxServicePort**. También debe habilitar las estadísticas en el archivo de propiedades del servidor. Los servidores de catálogos no son detectados por este filtro.
   - Si desea información sobre cómo iniciar los servidores con la opción **-jmxServicePort**, consulte “Script startOgServer” en la página 235.
   - Si desea más información sobre cómo habilitar las estadísticas en el archivo de propiedades de servidor, consulte la información sobre la propiedad **statsSpec** en “Archivo de propiedades de servidor” en la página 197.
2. Coloque el archivo `extremescale-plugin.xml` y el archivo `wxshyperic.jar` en los servidores apropiados de plug-in de servidor y agente de la configuración de Hyperic. Para integrar con Hyperic, las instalaciones de cliente y de servidor deben tener acceso a los archivos de plug-in y JAR (Java Archive). Aunque el servidor puede intercambiar dinámicamente configuraciones, debe completar la integración antes de iniciar cualquiera de los servicios.

a. Coloque el archivo `extremescale-plugin.xml` en el directorio `plugin` del servidor, que está en la siguiente ubicación:

   `hyperic_home/server_home/hq-engine/server/default/deploy/hq.ear/hq-plugins`

b. Coloque el archivo `extremescale-plugin.xml` en el directorio `plugin` del agente, que se encuentra en la siguiente ubicación:

   `agent_home/bundles/gent-4.0.2-939/pdk/plugins`

c. Coloque el archivo `wshyperic.jar` en el directorio `lib` del agente, que está en la siguiente ubicación

   `agent_home/bundles/gent-4.0.2-939/pdk/lib`

3. Configure el agente. El archivo `agent.properties` sirve como un punto de configuración para el tiempo de ejecución del agente. Esta propiedad está en el directorio *agent_home*`/conf`. Las siguientes claves son opcionales, pero son importantes para el plug-in de eXtreme Scale:
   - `autoinventory.defaultScan.interval.millis=<`*tiempo_en_milisegundos*`>`

     Establece el intervalo en milisegundos entre los descubrimientos de agente.
   - `log4j.logger.org.hyperic.hq.plugin.extremescale.XSServerDetector=DEBUG`

     : Habilita las sentencias de depuración verbosa desde el plug-in de eXtreme Scale.
   - `username=<`*nombre_usuario*`>`: establece el nombre de usuario de JMX (Java Management Extensions) si la seguridad está habilitada.
   - `password=<`*contraseña*`>`: establece la contraseña de JMX si la seguridad está habilitada.
   - `sslEnabled=<`*true*`|`*false*`>`: indica al plug-in si debe utilizar o no SSL (Secure Sockets Layer). El valor es `false` de forma predeterminada.
   - `trustPath=<`*vía_acceso*`>`: establece la vía de acceso de confianza para la conexión SSL.
   - `trustType=<`*tipo*`>`: establece el tipo de confianza para la conexión SSL.
   - `trustPass=<`*contraseña*`>`: establece la contraseña de confianza para la conexión SSL.
4. Inicie el descubrimiento del agente. Los agentes de Hyperic envían información de descubrimientos y métricas al servidor. Utilice el servidor para personalizar vistas de datos y objetos de inventario lógico de grupo para generar información útil. Después de que el servidor está disponible, debe ejecutar el script de inicio o iniciar el servicio de Windows para el agente:
   - Linux `agent_home/bin/hq-agent.sh start`
   - Windows Inicie el agente con el servicio de Windows.

   Después de iniciar los agentes, los servidores se detectan y los grupos se configuran. Puede iniciar la sesión en la consola de servidor y elegir qué recursos añadir a la base de datos de inventario para el servidor. La consola del servidor está en el siguiente URL de forma predeterminada: `http://<`*nombre_host_servidor*`>:7080/`
5. Supervise los servidores con la consola de Hyperic. Después de que se añadan los servidores al modelo de inventario, sus servicios ya no son necesarios.
   - **Vista del panel de instrumentos**: cuando visualizó los sucesos de detección de recursos, había iniciado la sesión en la vista del panel de instrumentos principal. La vista del panel de instrumentos es una vista genérica que hace

las veces de centro de mensajes que se puede personalizar. Puede exportar gráficos u objetos de inventario a este panel de instrumentos principal.

- **Vista de recursos**: puede consultar y ver todo el modelo de inventario desde esta página. Una vez que se han añadido los servicios, podrá ver cada uno de los servidores de eXtreme Scale etiquetados y listados correctamente bajo la sección de servidores. Puede pulsar los servidores individuales para ver las métricas básicas.

6. Vea todo el inventario del servidor en la página Ver recurso. En esta página, puede seleccionar varios servidores ObjectGrid y agruparlos juntos. Tras agrupar un conjunto de recursos, sus métricas comunes se pueden representar en un gráfico solapándose para mostrar las coincidencias y las diferencias entre los miembros del grupo. Para mostrar una coincidencia, seleccione las métricas en la pantalla del grupo de servidores. La métrica se visualiza en el área de gráficos. Para mostrar una coincidencia para todos los miembros del grupo, pulse el nombre de métrica subrayada. Puede exportar cualquiera de las gráficas, vistas de nodo y gráficas comparativas al panel de instrumentos principal con el menú **Herramientas**.

Conceptos relacionados

"Herramientas del proveedor" en la página 290
WebSphere eXtreme Scale se puede supervisar utilizando varias soluciones populares de supervisión empresarial. Los agentes de plug-in se incluyen para IBM Tivoli Monitoring e Hyperic HQ, que supervisan WebSphere eXtreme Scale utilizando los beans de gestión a los que se puede acceder públicamente. CA Wily Introscope utiliza la instrumentación de métodos Java para capturar las estadísticas.

"Visión general de las estadísticas" en la página 261
Las estadísticas de WebSphere eXtreme Scale se calculan basándose en un árbol de estadísticas internas. La API StatsAccessor, los módulos PMI (Performance Monitoring Infrastructure) y la API MBean se generan desde el árbol interno.

Registros y rastreo

Puede utilizar los registros y el rastreo para supervisar y resolver problemas del entorno. Los registros están en distintas ubicaciones en función de su configuración. Es posible que sea necesario proporcionar el rastreo para un servidor cuando se trabaja con el servicio de soporte IBM.

Registros con WebSphere Application Server

Consulte WebSphere Application Server Information Center para obtener más información.

Registros con WebSphere eXtreme Scale en un entorno autónomo

Con servidores de catálogo autónomo y de contenedor, establezca la ubicación de los registros y toda especificación de rastreo. Los registros del servidor de catálogo se encuentran en la ubicación en la que ejecutó el mandato de inicio de servidor.

Establecimiento de la ubicación de registro para los servidores de contenedor

De forma predeterminada, los registros de un contenedor están en el directorio en el que se ha ejecutado el mandato de servidor. Si inicia los servidores en el directorio `<inicio_eXtremeScale>/bin`, los registros y los archivos de rastreo se encuentran en el directorio `logs/<nombre_servidor>` del directorio `bin`. Para

especificar una ubicación alternativa de los registros de un servidor de contenedor, cree un archivo de propiedades, como el archivo `server.properties`, con el siguiente contenido:

```
workingDirectory=<directorio>
traceSpec=
systemStreamToFileEnabled=true
```

La propiedad workingDirectory es el directorio raíz para los registros y el archivo de rastreo opcional. WebSphere eXtreme Scale crea un directorio con el nombre del servidor de contenedor con un archivo `SystemOut.log`, un archivo `SystemErr.log` y un archivo de rastreo si el rastreo se ha habilitado con la opción traceSpec. Para utilizar un archivo de propiedades durante el inicio del contenedor, utilice la opción **-serverProps** y proporcione la ubicación del archivo de propiedades de servidor.

Consulte "Inicio de servidores de WebSphere eXtreme Scale autónomos" en la página 228 y "Script startOgServer" en la página 235 para obtener más información.

Los mensajes de información común que se deben buscar en el archivo `SystemOut.log` son mensajes de confirmación de inicio. Si desea más información sobre un mensaje específico, consulte "Mensajes" en la página 335.

Rastreo con WebSphere Application Server

Consulte WebSphere Application Server Information Center para obtener más información.

Rastreo en el servicio de catálogo

Puede establecer el rastreo en un servicio de catálogo utilizando los parámetros **-traceSpec** y **-traceFile** durante el inicio del servicio de catálogo. Por ejemplo:

```
startOgServer.sh catalogServer -traceSpec
ObjectGridPlacement=all=enabled -traceFile
/home/user1/logs/trace.log
```

Si inicia el servicio de catálogo en el directorio <*inicio_eXtremeScale*>/`bin`, los registros y los archivos de rastreo estarán en un directorio `logs/`<*nombre_servicio_catálogo*> en el directorio `bin`. Consulte "Inicio del servicio de catálogo en un entorno autónomo" en la página 229 si desea más detalles sobre cómo iniciar un servicio de catálogo.

Rastreo en un servidor de contenedor autónomo

Puede habilitar el rastreo en un servidor de contenedor de dos formas. Puede crear un archivo de propiedades de servidor tal como se explica en la sección de registros, o puede habilitar el rastreo utilizando la línea de mandatos durante el inicio. Para habilitar el rastreo de contenedor con un archivo de propiedades de servidor, actualice la propiedad **traceSpec** con la especificación de rastreo necesaria. Para habilitar el rastreo de contenedor utilizando parámetros de inicio, utilice los parámetros **-traceSpec** y **-traceFile**. Por ejemplo:

```
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
```

Si inicia el servidor en el directorio <*inicio_eXtremeScale*>/`bin`, los registros y archivos de rastreo se encuentran en los directorios `logs/`<*nombre_servidor*> en el directorio `bin` Consulte "Inicio de procesos de contenedor" en la página 232 si

desea más detalles sobre cómo iniciar un proceso de contenedor.

Rastreo con la interfaz ObjectGridManager

Otra opción es establecer el rastreo durante la ejecución de una interfaz ObjectGridManager. Si se establece el rastreo en una interfaz ObjectGridManager, se puede utilizar para obtener el rastreo en un cliente de eXtreme Scale, mientras se conecta a eXtreme Scale y confirma transacciones. Para establecer el rastreo en una interfaz ObjectGridManager, proporcione una especificación de rastreo y un registro de rastreo.

```
ObjectGridManager manager = ObjectGridManagerFactory.getObjectGridManager();
...
manager.setTraceEnabled(true);
manager.setTraceFileName("logs/myClient.log");
manager.setTraceSpecification("ObjectGridReplication=all=enabled");
```

Habilitación del rastreo con el programa de utilidad xsadmin

Para habilitar el rastreo con el programa de utilidad xsadmin, utilice la opción **setTraceSpec**. Utilice el programa de utilidad xsadmin para habilitar el rastreo en un entorno autónomo durante la ejecución, en lugar de durante el arranque. Puede habilitar el rastreo en todos los servidores y servicios de catálogo o puede filtrar los servidores basándose en el nombre de ObjectGrid, etc. Por ejemplo, para habilitar el rastreo de ObjectGridReplication con acceso al servidor del servicio de catálogo, ejecute:

```
<inicio_eXtremeScale>/bin>xsadmin.bat -setTraceSpec "ObjectGridReplication=all=enabled"
```

También puede inhabilitar el rastreo estableciendo la especificación de rastreo en `*=all=disabled.`

Consulte “Utilización del programa de utilidad de ejemplo xsAdmin” en la página 287 si desea más información.

Archivos y directorio ffdc

Los archivos FFDC sirven para que el servicio de soporte de IBM ayude a realizar la depuración. El servicio de soporte de IBM puede solicitar estos archivos si se produce un problema.

Estos archivos están en un directorio denominado, `ffdc`, y contienen archivos que se parecen al siguiente:

```
server2_exception.log
server2_20802080_07.03.05_10.52.18_0.txt
```

Conceptos relacionados

"Inicio y parada de servidores eXtreme Scale seguros" en la página 327
A menudo, los servidores necesitan ser seguros para el entorno de despliegue, que requiere una configuración específica.

Tareas relacionadas

"Inicio de servidores de WebSphere eXtreme Scale autónomos" en la página 228
Cuando se ejecuta una configuración de WebSphere eXtreme Scale autónoma, el entorno está formado por servidores de catálogo, servidores de contenedor y procesos de cliente de eXtreme Scale. Debe configurar e iniciar manualmente estos procesos.

"Inicio del servicio de catálogo en un entorno autónomo" en la página 229
Debe iniciar el servicio de catálogo manualmente cuando utilice un entorno distribuido de WebSphere eXtreme Scale que no ejecute WebSphere Application Server.

"Inicio de procesos de contenedor" en la página 232
Puede iniciar eXtreme Scale desde la línea de mandatos, utilizando una topología de despliegue, o utilizando un archivo `server.properties`.

Opciones de rastreo

Puede habilitar el rastreo para proporcionar información sobre el entorno al servicio de soporte de IBM.

Sobre el rastreo

El rastreo de WebSphere eXtreme Scale se divide en varios componentes distintos. De forma similar al rastreo de WebSphere Application Server, puede especificar el nivel de rastreo que se debe utilizar. Los niveles comunes de rastreo son: all, debug, entryExit y event.

A continuación se muestra una serie de rastreo de ejemplo:

```
ObjectGridComponent=level=enabled
```

Puede concatenar valores de rastreo. Utilice el símbolo `*` (asterisco) para especificar un valor comodín como, por ejemplo, `ObjectGrid*=all=enabled`. Si necesita proporcionar un rastreo al servicio de soporte de IBM, se solicita una serie de rastreo específica. Por ejemplo, si hay un problema con la réplica, se puede solicitar la serie de rastreo `ObjectGridReplication=debug=enabled`.

Especificación de rastreo

ObjectGrid
Motor de memoria caché principal general.

ObjectGridCatalogServer
Servicio de catálogo general.

ObjectGridChannel
Comunicaciones de topología de despliegue estática.

ObjectgridCORBA
Comunicaciones de topología de despliegue dinámica.

ObjectGridDataGrid
API de AgentManager.

ObjectGridDynaCache
El proveedor de la memoria caché dinámica de WebSphere eXtreme Scale.

ObjectGridEntityManager
La API de EntityManager. Utilícela con la opción Projector.

ObjectGridEvictors
Desalojadores incorporados de ObjectGrid.

ObjectGridJPA
Cargadores JPA (Java Persistence API).

ObjectGridJPACache
Plug-ins de memoria caché JPA.

ObjectGridLocking
Gestor de bloqueos de entradas de memoria caché de ObjectGrid.

ObjectGridMBean
Beans de gestión.

ObjectGridPlacement
Servicio de colocación de fragmentos de servidor de catálogo.

ObjectGridQuery
Consulte de ObjectGrid.

ObjectGridReplication
Servicio de réplica.

ObjectGridRouting
Detalles de direccionamiento de cliente/servidor.

ObjectGridSecurity
Rastreo de seguridad.

ObjectGridStats
Estadísticas de ObjectGrid.

ObjectGridStreamQuery
La API de consulta de secuencia.

ObjectGridWriteBehind
Escritura diferida de ObjectGrid

Projector
El motor dentro de la API de EntityManager.

QueryEngine
El motor de consulta para la API de consulta de objetos y la API de consulta de EntityManager.

QueryEnginePlan
Diagnósticos del plan de consulta.

Capítulo 9. Protección del entorno de despliegue

Para proteger los datos de WebSphere eXtreme Scale, eXtreme Scale puede integrarse con proveedores de seguridad externos.

WebSphere eXtreme Scale puede integrarse con una implementación de seguridad externa. Esta implementación externa debe proporcionar servicios de autenticación y autorización para eXtreme Scale. eXtreme Scale tiene puntos de plug-in para integrarse con una implementación de seguridad. eXtreme Scale se ha integrado correctamente con los siguientes componentes:

- Lightweight Directory Access Protocol (LDAP)
- Kerberos
- Seguridad de ObjectGrid
- Tivoli Access Manager
- JAAS (Java Authentication and Authorization Service)

eXtreme Scale utiliza el proveedor de seguridad para las siguientes tareas:

- Autenticación de clientes en servidores.
- Autorización de clientes para acceder a determinados artefactos de eXtreme Scale o para especificar qué puede hacerse con los artefactos de eXtreme Scale.

eXtreme Scale tiene los siguientes tipos de autorizaciones:

Autorización de correlaciones
: Los clientes o grupos pueden estar autorizados para realizar operaciones de inserción, lectura, actualización o supresión en correlaciones.

Autorización de ObjectGrid
: Los clientes o grupos pueden estar autorizados para realizar consultas de objeto o entidad en cuadrículas de objetos.

Autorización de agentes de DataGrid
: Los clientes o grupos pueden estar autorizados para permitir que se desplieguen los agentes DataGrid en un ObjectGrid.

Autorización de correlaciones del lado de servidor
: Los clientes o grupos pueden estar autorizados para duplicar una correlación de servidor en el lado del cliente o para crear un índice dinámico en la correlación del servidor.

Autorización de administración
: Los clientes o grupos pueden estar autorizados para realizar tareas de administración.

Nota: Si ya tenía habilitada la seguridad para el programa de fondo, recuerde que estos valores de seguridad ya no serán suficiente para proteger los datos. Los valores de seguridad de la base de datos u otro almacén de datos no se transfiere en absoluto a la memoria caché. Debe proteger de forma separada los datos que ahora están almacenados en la memoria caché utilizando el mecanismo de seguridad de eXtreme Scale, incluido la seguridad de autenticación, autorización y nivel de transporte.

Seguridad de la cuadrícula

La seguridad de la cuadrícula de WebSphere eXtreme Scale garantiza que el servidor que se una tenga las credenciales correctas, de modo que un servidor malintencionado no puede unirse a la cuadrícula. Utiliza un mecanismo de serie secreta compartida para este propósito.

Todos los servidores de WebSphere eXtreme Scale, incluidos los servidores de catálogo, acuerdan una serie secreta compartida. Cuando un servidor se une a la cuadrícula, se ve obligado a presentar la serie secreta. Si la serie secreta del servidor que se une coincide con la serie del servidor presidente o el servidor de catálogo, se acepta el servidor que se une. Si la serie no coincide, se rechaza la solicitud de unión.

El envío de un texto visible no es seguro. La infraestructura de seguridad de WebSphere eXtreme Scale proporciona un plug-in de gestor de señales seguras para permitir al servidor proteger este secreto antes de enviarlo. Debe decidir cómo implementar la operación segura. WebSphere eXtreme Scale proporciona una implementación directa, en la que la operación segura se implementa para cifrar y firmar el secreto.

La serie secreta se establece en el archivo `server.properties`. Consulte “Archivo de propiedades de servidor” en la página 197 si desea más información sobre la propiedad authenticationSecret.

Plug-in SecureTokenManager

Un plug-in de gestor de señales seguras se representa mediante la interfaz com.ibm.websphere.objectgrid.security.plugins.SecureTokenManager.

Si desea más información sobre el plug-in SecureTokenManager, consulte la documentación de la API SecureTokenManager.

El método generateToken(Object) toma un objeto y, a continuación, genera una señal que no los otros no pueden entender. El método verifyTokens(byte[]) realiza el proceso inverso: el método convierte la señal en el objeto original.

Una implementación sencilla de SecureTokenManager utiliza un algoritmo de codificación sencillo, como un algoritmo exclusivo o (XOR), para codificar el objeto en un formato serializado y, a continuación, utilizar el algoritmo de decodificación correspondiente para descifrar la señal. Esta implementación no es segura.

WebSphere eXtreme Scale proporciona una implementación disponible de forma inmediata para esta interfaz.

La implementación predeterminada utiliza un par de claves para firmar y verificar la firma, y utiliza una clave secreta para cifrar el contenido. Cada servidor tiene un almacén de claves de tipo JCKES donde se almacena el par de claves, una clave privada y una clave pública, y un clave secreta. El almacén de claves tiene que ser de tipo JCKES para poder almacenar las claves secretas.

Estas claves se utilizan para cifrar y firmar o verificar la serie secreta en el envío. Además, la señal se asocia con un tiempo de caducidad. En el extremo receptor, los datos se verifican, se descifran y se comparan con la serie secreta del receptor. Los protocolos de comunicación SSL (Secure Sockets Layer) no son obligatorios para la autenticación entre un par de servidores porque las claves privadas y públicas

sirven para ese mismo propósito. No obstante, si la comunicación del servidor no está cifrada, los datos podrían robarse con sólo observar la comunicación. Como la señal caduca pronto, la amenaza de ataque de reproducción se minimiza. Esta posibilidad disminuye en gran medida si todos los servidores se despliegan detrás de un cortafuegos.

La desventaja de este enfoque es que los administradores de WebSphere eXtreme Scale deben generar claves y transportarlas a todos los servidores, que puede provocar una violación de seguridad durante el transporte.

Configuración

Consulte Propiedades de servidor si desea más información sobre las propiedades que utiliza para configurar el gestor de señales seguras.

Habilitación de la seguridad local

WebSphere eXtreme Scale proporciona varias puntos finales de seguridad para integrar mecanismos personalizados. En el modelo de programación local, la principal función se seguridad es la autorización, que no tiene soporte de autenticación. Debe realizar la autenticación de forma independiente desde la autenticación que ya existe de WebSphere Application Server. No obstante, se proporcionan plug-ins con el fin de obtener y validar objetos Subject.

Las secciones siguientes describen las dos formas en las que puede habilitar la seguridad local:

Puede utilizar el archivo XML de ObjectGrid para definir una ObjectGrid y habilitar la seguridad para dicha ObjectGrid. El archivo `secure-objectgrid-definition.xml` que se utiliza en la aplicación empresarial de ObjectGridSample se muestra en el siguiente ejemplo. Establezca el atributo securityEnabled en `true` para habilitar la seguridad.

```
<objectGrids>
    <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
          authorizationMechanism="AUTHORIZATION_MECHANISM_JAAS">
    ...
</objectGrids>
```

También puede habilitar la seguridad a través de programas. Para crear una ObjectGrid utilizando el método ObjectGrid.setSecurityEnabled, llame al siguiente método en la interfaz ObjectGrid:

```
/**
 * Habilitar la seguridad ObjectGrid
 */
void setSecurityEnabled();
```

Si desea más información, consulte la documentación de la API.

Autenticación de cliente de aplicaciones

La autenticación del cliente de aplicaciones consiste en la habilitación de la seguridad de cliente-servidor y la autenticación de credenciales, y en la configuración de un autenticador y un generador de credenciales de sistema.

Habilitación de la seguridad cliente/servidor

Debe habilitar la seguridad tanto en el cliente, como en el servidor, para autenticarse correctamente con ObjectGrid.

Habilitar seguridad de cliente

WebSphere eXtreme Scale proporciona un archivo de ejemplo de propiedad de cliente, el archivo `sampleClient.properties`, en el directorio *`WAS_HOME`*`/optionalLibraries/ObjectGrid/properties` para una instalación de WebSphere Extended Deployment o el directorio `/ObjectGrid/properties` en una instalación de servidor combinada. Puede modificar este archivo de plantilla al añadir valores adecuados. Establezca la propiedad securityEnabled en el archivo `objectgridClient.properties` en `true`. La propiedad securityEnabled indica si la seguridad está habilitada. Cuando un cliente se conecta a un servidor, el valor en el cliente y en el servidor se deben establecer ambos en `true` o ambos en `false`. Por ejemplo, si el servidor conectado está habilitado, el valor de propiedad se debe establecer en `true` en el cliente para que el cliente se conecte al servidor.

La interfaz com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration representa el archivo `security.ogclient.props`. Puede usar la API pública com.ibm.websphere.objectgrid.security.config.ClientSecurityConfigurationFactory para crear una instancia de esta interfaz con valores predeterminados, o puede crear una instancia al pasar el archivo de propiedades de seguridad de cliente ObjectGrid. El archivo `security.ogclient.props` contiene otras propiedades. Consulte la documentación de la API ClientSecurityConfiguration y la documentación de la API ClientSecurityConfigurationFactory si desea más detalles.

Habilitar la seguridad del servidor

Para habilitar la seguridad en el lado del servidor, puede establecer la propiedad securityEnabled del archivo `security.xml` en `true`. Utilice un archivo XML de descriptor de seguridad para especificar la configuración de seguridad de la cuadrícula para aislar la configuración de seguridad de nivel de cuadrícula de la configuración sin seguridad.

Habilitación de la autenticación de credenciales

Después de que el cliente de eXtreme Scale recupere el objeto Credential utilizando el objeto CredentialGenerator, el objeto Credential se envía junto con la petición de cliente al servidor eXtreme Scale. El servidor autentica el objeto Credential antes de procesar la solicitud. Si el objeto Credential se ha autenticado correctamente, se devuelve un objeto Subject para representar este objeto Credential. Este objeto Subject se utiliza para autorizar la petición.

Establezca credentialAuthentication en los archivos de propiedades de cliente y de servidor para habilitar la autenticación de credenciales. Si desea más información, consulte “Archivo de propiedades de cliente” en la página 203 y “Archivo de propiedades de servidor” en la página 197.

La siguiente tabla proporciona qué mecanismos de autenticación utilizar bajo distintos valores.

Tabla 9. Autenticación de credenciales bajo los valores de cliente y servidor

Autenticación de credenciales de cliente	Autenticación de credenciales de servidor	Resultado
No	Nunca	Inhabilitado
No	Soportado	Inhabilitado
No	Necesario	Caso de error

Tabla 9. Autenticación de credenciales bajo los valores de cliente y servidor (continuación)

Autenticación de credenciales de cliente	Autenticación de credenciales de servidor	Resultado
Soportado	Nunca	Inhabilitado
Soportado	Soportado	Habilitado
Soportado	Necesario	Habilitado
Necesario	Nunca	Caso de error
Necesario	Soportado	Habilitado
Necesario	Necesario	Habilitado

Configuración de una autenticador

El servidor eXtreme Scale utiliza el plug-in Authenticator para autenticar el objeto Credential. Una implementación de la interfaz Authenticator obtiene el objeto Credential y, después, lo autentica en un registro de usuarios, por ejemplo, un servidor LDAP (Lightweight Directory Access Protocol), etc. eXtreme Scale no proporciona una configuración de registro. Se debe implementar una conexión a un registro de usuarios y autenticarla en este plug-in.

Por ejemplo, una implementación de Authenticator extrae el ID de usuario y la contraseña de la credencial, los utiliza para conectarse y validar un servidor LDAP y crea un objeto Subject como resultado de la autenticación. La implementación puede utilizar los módulos de inicio de sesión JAAS (Java Authentication and Authorization Service). Como resultado de la autenticación, se devuelve un objeto Subject.

Puede configurar el autenticador en el archivo XML de descriptor de seguridad, tal como se indica en el siguiente ejemplo:

```
<?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>
```

Utilice la opción –clusterSecurityFile al iniciar un servidor seguro para establecer el archivo XML de seguridad. Consulte la guía de aprendizaje de seguridad de Java SE en *Visión general del producto* si desea más información.

Configuración de un generador de credenciales del sistema

El generador de credenciales del sistema se utiliza para representar una fábrica de la credencial del sistema.Una credencial del sistema es similar a una credencial del administrador. Puede configurar el elemento SystemCredentialGenerator en el XML de la seguridad del catálogo, tal como se muestra en el siguiente ejemplo:

```
<systemCredentialGenerator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator">
          <property name="properties" type="java.lang.String" value="manager manager1" description="username password" />
    </systemCredentialGenerator>
```

Por motivos de demostración, el nombre de usuario y la contraseña se almacenan en texto visible. No almacene el nombre de usuario y la contraseña en texto visible en un entorno de producción.

WebSphere eXtreme Scale proporciona un generador de credenciales del sistema predeterminado, que utiliza las credenciales del servidor. Si no especifica explícitamente el generador de credenciales del sistema, se utiliza este generador de credenciales del sistema predeterminado.

Autorización de cliente de aplicaciones

La autorización del cliente de aplicaciones consta de clases de permisos de ObjectGrid, mecanismos de autorización, un periodo de comprobación de permisos y un acceso sólo por parte de la la autorización del creador.

Para eXtreme Scale, la autorización se basa en el objeto Subject y los permisos. El producto soporta dos tipos de mecanismos de autorización: Java Authentication and Authorization Service (JAAS) y la autorización personalizado.

Clases de permiso ObjectGrid

La autorización se basa en permisos. Existen cuatro tipos diferentes de clases de permiso del modo siguiente.

- La clase MapPermission representa los permisos para acceder a los datos en correlaciones de ObjectGrid.
- La clase ObjectGridPermission representa los permisos para acceder a ObjectGrid.
- La clase ServerMapPermission representa los permisos para acceder a correlaciones de ObjectGrid en el servidor desde un cliente.
- La clase AgentPermission representa los permisos para iniciar un agente en el servidor.

Si desea más información sobre las API y los permisos asociados, consulte el tema sobre la programación de autorización de cliente en *Guía de programación*.

Período de comprobación de permisos

eXtreme Scale soporta el almacenamiento en memoria caché de los resultados de la comprobación de permisos de correlación con finalidades de rendimiento. Sin este mecanismo, cuando se llama a un método que aparece en la lista de métodos y sus permisos necesarios, el tiempo de ejecución llama al mecanismo de autorización configurada para autorizar el acceso. Con este período de comprobación de permisos establecido, el mecanismo de autorización se llama periódicamente en función del período de comprobación de permisos.

La información de autorización de permisos se basa en el objeto Subject. Cuando un cliente intenta acceder a los métodos, el tiempo de ejecución de eXtreme Scale consulta la memoria caché en función del objeto Subject. Si el objeto no se encuentra en la memoria caché, el tiempo de ejecución comprueba los permisos concedidos para este objeto Subject, y luego almacena los permisos en una memoria caché.

El período de comprobación de permisos debe definirse antes de inicializar ObjectGrid. El período de comprobación de permisos puede configurarse de dos modos:

Puede utilizar el archivo XML de ObjectGrid para definir un ObjectGrid y establecer el periodo de comprobación de permisos. En el siguiente ejemplo, el periodo de comprobación de permisos se establece en 45 segundos:

```
<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>
```

Si desea crear un ObjectGrid con API, llame al siguiente método para establecer el periodo de comprobación de permisos. Este método sólo puede llamarse antes de inicializar la instanciel ObjectGrid. Este método se aplica sólo al modelo de programación local de eXtreme Scale cuando cree una instancia directamente de ObjectGrid.

```
/**
 * Este método toma un único parámetro que indica con qué frecuencia
 * desea comprobar el permiso utilizado para permitir un acceso de cliente. Si el
 * parámetro es 0 cada llamada única get/put/update/remove/evict
 * solicita al mecanismo de autorización, autorización JAAS o personalizada,
 * comprobar si el objeto Subject actual tiene permiso. Esto podría ser
 * muy costoso desde el punto de vista del rendimiento en función de
 * la implementación de autorización, pero si necesita comprobar el
 * mecanismo de autorización, establezca el parámetro en 0.
 * De forma alternativa, si el parámetro es > 0, indica el número
 * de segundos que tarda en almacenar en la memoria caché un conjunto de
permisos antes de volver al
 * mecanismo de autorización para que los actualice. Este valor proporciona un
 * mejor rendimiento, pero si los permisos del programa de fondo
 * se cambian durante este tiempo, ObjectGrid puede
 * permitir o denegar el acceso aunque el proveedor de seguridad
 * del programa de fondo se haya modificado.
 *
 * @establecer el parámetro period para el período de comprobación de permisos
en segundos.
 */
void setPermissionCheckPeriod(int period);
```

Autorización de sólo acceso de creador

La autorización de sólo acceso de creador garantiza que sólo el usuario (representado por los objetos Principal asociados a él) que inserta la entrada en la correlación ObjectGrid pueda acceder (leer, actualizar, invalidar y eliminar) a la entrada.

El modelo de autorización de la correlación ObjectGrid existente se basa en el tipo de acceso, no en las entradas de datos. En otras palabras, un usuario tiene un tipo determinado de acceso (leer, grabar, insertar, suprimir o invalidar) para todos los datos de la correlación o para ninguno. No obstante, eXtreme Scale no autoriza a los usuarios la entrada individual de los datos. Esta característica ofrece una nueva manera de autorizar a los usuarios las entradas de datos.

En un escenario donde diferentes usuarios pueden acceder a distintos conjuntos de datos, este modelo puede ser de utilidad. Cuando el usuario carga los datos del almacén persistente en las correlaciones ObjectGrid, el acceso puede autorizarse desde el almacén persistente. En este caso, no es necesario realizar otra autorización en la capa de correlación ObjectGrid. Sólo debe asegurarse de que la persona que carga los datos en la correlación pueda acceder a ella mediante la habilitación de la característica de sólo acceso de creador.

Existen tres modalidades diferentes de sólo acceso de creador:

disabled
La característica de sólo acceso de creador está inhabilitada.

complement
La característica de sólo acceso de creador está habilitada para complementar la autorización de correlaciones. En otras palabras, la autorización de correlaciones y, también, la característica de sólo acceso de creador entran en vigor. Por lo tanto, puede limitar las operaciones a los datos. Por ejemplo, el creador no puede invalidar los datos.

supersede
La característica de sólo acceso de creador está habilitada para reemplazar la autorización de correlaciones. En otras palabras, la característica de sólo acceso de creador reemplaza la autorización de correlaciones; no se produce ninguna autorización de correlaciones.

Puede configurar esta característica de dos modos:

Puede utilizar el archivo XML de ObjectGrid para definir un ObjectGrid y establecer la modalidad de sólo acceso de creador en `disabled`, `complement` o `supersede`, tal como se indica en el siguiente ejemplo:

```
<objectGrids>
    <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
         accessByCreatorOnlyMode="supersede"
        <bean id="TransactionCallback"
              classname="com.ibm.websphere.samples.objectgrid.HeapTransactionCallback" />
    ...
</objectGrids>
```

Si desea crear un ObjectGrid mediante programa, puede llamar al siguiente método para establecer la modalidad de sólo acceso de creador. La llamada a este método sólo se aplica al modelo de programación de eXtreme Scale local cuando se crea directamente una instancia de ObjectGrid:

```
/**
 * Establezca la modalidad de sólo acceso de creador.
 * Si habilita esta modalidad se asegura de que sólo el usuario (representado
 * por los principales asociados a éste), que inserta el registro en la correlación,
 * pueda acceder (leer, actualizar, invalidar y eliminar) al registro.
 * La modalidad de sólo acceso de creador puede inhabilitarse, o puede complementar
 * el modelo de autorización ObjectGrid, o puede reemplazar el modelo de autorización
 * ObjectGrid. El valor predeterminado es la modalidad inhabilitada:
 * {@link SecurityConstants#ACCESS_BY_CREATOR_ONLY_DISABLED}.
 * @ver SecurityConstants#ACCESS_BY_CREATOR_ONLY_DISABLED
 * @ver SecurityConstants#ACCESS_BY_CREATOR_ONLY_COMPLEMENT
 * @ver SecurityConstants#ACCESS_BY_CREATOR_ONLY_SUPERSEDE
 *
 * @establecer el parámetro accessByCreatorOnlyMode para la modalidad
de sólo acceso de creador.
 *
 * @desde WAS XD 6.1 FIX3
*/
void setAccessByCreatorOnlyMode(int accessByCreatorOnlyMode);
```

Con el propósito de ilustrar con más detalle, considere un escenario en el que una cuenta de correlaciones de ObjectGrid está en una cuadrícula de banca, y Manager1 y Employee1 son los dos usuarios. La política de autorización de eXtreme Scale otorga todos los permisos de acceso a Manager1, pero sólo otorga un permiso de acceso de lectura a Employee1. La política JAAS para la autorización de correlación ObjectGrid se muestra en el siguiente ejemplo:

```
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"
};
```

Considere cómo la modalidad de sólo acceso de creador afecta a la autorización:

- **disabled** Si la característica de sólo acceso de creador está inhabilitada, la autorización de correlaciones no cambia. El usuario ″Manager1″ puede acceder a todos los datos de la correlación de cuenta ″account″. El usuario ″Employee1″ puede leer todos los datos de la correlación pero no puede actualizarlos, invalidarlos ni eliminar ningún dato de la correlación.
- **complement** Si la característica de sólo acceso de creador está habilitada con la opción complementaria ″complement″, entrarán en vigor la autorización de correlaciones y la autorización de sólo acceso de creador. El usuario ″Manager1″ puede acceder a los datos de la correlación de cuenta ″account″, pero sólo si el usuario los ha cargado en la correlación. El usuario ″Employee1″ puede leer los datos de la correlación de cuenta ″account″, pero sólo si ese usuario los ha cargado en la correlación. No obstante, este usuario no puede actualizar, invalidar ni eliminar ningún dato en la correlación.
- **supersede** Si la característica de sólo acceso de creador está habilitada con la opción de reemplazar ″supersede″, no se aplicará la autorización de correlaciones. La autorización de sólo acceso de creador será la única política de autorización. El usuario ″Manager1″ tiene el mismo privilegio que en la modalidad ″complement″: este usuario puede acceder a los datos de la correlación de cuenta ″account″ sólo si ese mismo usuario ha cargado los datos en la correlación. No obstante, el usuario ″Employee1″ ahora tiene acceso completo a los datos de la correlación ″account″ si este usuario los ha cargado en la correlación. En otras palabras, la política de autorización definida en la política JAAS (Java Authentication and Authorization Service) no se aplicará.

Transport Layer Security (TLC) y Secure Sockets Layer (SSL)

WebSphere eXtreme Scale 0soporta TCP/IP y TLS/SSL (Transport Layer Security/Secure Sockets Layer) para la comunicación segura entre el cliente y el servidor en los modelos de despliegue dinámico.

TLS/SSL proporciona comunicación segura entre el cliente y el servidor. El mecanismo de comunicación que se utiliza depende del valor del parámetro transportType especificado en los archivos de configuración del cliente y del servidor.

Puede establecer la propiedad transportType en los siguientes archivos de configuración de cliente y servidor:

- Para establecer la propiedad en la configuración de seguridad de cliente, consulte “Archivo de propiedades de cliente” en la página 203.
- Para establecer la propiedad en la configuración de seguridad del servidor de contenedor, consulte “Archivo de propiedades de servidor” en la página 197.
- Para establecer la propiedad en la configuración de seguridad del servidor de catálogo, consulte “Archivo de propiedades de servidor” en la página 197.

Tabla 10. Protocolo de transporte a utilizar bajo los valores de transporte de cliente y de transporte de servidor

Propiedad transportType de cliente	Propiedad transportType de servidor	Protocolo resultante
TCP/IP	TCP/IP	TCP/IP
TCP/IP	Da soporte a SSL	TCP/IP
TCP/IP	Precisa SSL	Error
Da soporte a SSL	TCP/IP	TCP/IP
Da soporte a SSL	Da soporte a SSL	SSL (si SSL falla, TCP/IP)
Da soporte a SSL	Precisa SSL	SSL
Precisa SSL	TCP/IP	Error

Tabla 10. Protocolo de transporte a utilizar bajo los valores de transporte de cliente y de transporte de servidor (continuación)

Propiedad transportType de cliente	Propiedad transportType de servidor	Protocolo resultante
Precisa SSL	Da soporte a SSL	SSL
Precisa SSL	Precisa SSL	SSL

Cuando se utiliza SSL, se deben proporcionar los parámetros de configuración de SSL tanto en el lado del cliente, como en el lado del servidor. En un entorno Java SE, la configuración de SSL se realiza en los archivos de propiedad de cliente o servidor. Si el cliente o el servidor está en WebSphere Application Server, podrá utilizar el soporte de seguridad de transportes de WebSphere Application Server para configurar los parámetros de SSL.

Configuración del archivo `orb.properties` para el soporte de seguridad de transportes

Puede utilizar TLS/SSL cuando la propiedad transportType tiene un valor de SSL soportado.

Para soportar un transporte seguro en un entorno de Java Platform, Standard Edition, debe modificar el archivo "Archivo de propiedades ORB" en la página 207 para incluir las siguientes propiedades:

```
# Propiedades de 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

# Plugins de WS
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

# Interceptores de WS
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.objectgrid.corba.ObjectGridInitializer
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ISecurityLocalObjectBaseL13Impl.SecurityComponentFactory

# Propiedades de plugins y ORB de WS
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
```

Configuración de parámetros SSL para los clientes de eXtreme Scale

Puede configurar los parámetros SSL para los clientes de las siguientes formas:

1. Cree un objeto com.ibm.websphere.objectgrid.security.config.SSLConfiguration utilizando la clase de fábrica com.ibm.websphere.objectgrid.security.config. ClientSecurityConfigurationFactory. Para obtener más información, consulte la documentación de la API ClientSecurityConfigurationFactory.
2. Configure los parámetros en el archivo `client.propeties` y, a continuación, utilice el método ClientSecurityConfigurationFactory.getClientSecurityConfiguration(String) para llenar la instancia del objeto.

Consulte la sección sobre las propiedades de cliente de seguridad en "Archivo de propiedades de cliente" en la página 203 para ver ejemplos de propiedades que puede establecer en un cliente.

Configuración de los parámetros SSL para servidores eXtreme Scale

Los parámetros SSL se configuran para los servidores que utilizan un archivo de propiedades de servidor como, por ejemplo, los ejemplos del archivo `server.properties` al que se hace referencia arriba. Este archivo de propiedades se puede pasar como un parámetro al iniciar un servidor eXtreme Scale. Si desea más información sobre los parámetros SSL que puede establecer para los servidores eXtreme Scale, consulte "Archivo de propiedades de servidor" en la página 197.

Soporte de seguridad de transporte en WebSphere Application Server

Cuando un cliente, un servidor de contenedor o un servidor de catálogo de eXtreme Scale se ejecuta en un proceso de WebSphere Application Server, la seguridad de transporte de eXtreme Scale es gestionada por los valores de transporte CSIV2 del servidor de aplicaciones. No debe utilizar las propiedades del cliente o servidor de eXtreme Scale para configurar los valores de SSL. Todos los valores de SSL se deben especificar en la configuración de WebSphere Application Server.

Nota:

Si realiza la configuración con valores de SSL con un archivo de propiedades, alterará temporalmente los valores existentes.

Referencia relacionada

"Referencia del archivo de propiedades" en la página 196
Los archivos de propiedades del servidor contienen valores para ejecutar los servidores de catálogo y los servidores de contenedor. Puede especificar un archivo de propiedades de servidor para una configuración autónoma o de WebSphere Application Server. Los archivos de propiedades de cliente contienen valores para el cliente.

"Archivo de propiedades de servidor" en la página 197
El archivo de propiedades de servidor contiene varias propiedades que definen distintos valores para el servidor como, por ejemplo, los valores de rastreo, el inicio de sesión y la configuración de seguridad. El servicio de catálogos y los servidores de contenedor utilizan el archivo de propiedades de servidor.

"Archivo de propiedades de cliente" en la página 203
Puede crear un archivo de propiedades basándose en los requisitos para los procesos de cliente de eXtreme Scale.

"Archivo de propiedades ORB" en la página 207
El archivo `orb.properties` se utiliza para pasar las propiedades utilizadas por el intermediario de solicitud de objetos (ORB) para modificar el comportamiento de transporte de la cuadrícula.

Autenticación de cuadrícula

El plug-in del gestor de señales seguras es otro plug-in utilizado para la autenticación del servidor. El plug-in del gestor de señales seguras está representado por la interfaz com.ibm.websphere.objectgrid.security.plugins.SecureTokenManager.

El método generateToken(Object) toma un objeto y, a continuación, genera una señal que los otros no pueden entender. El método verifyTokens(byte[]) realiza el proceso inverso: convierte la señal en el objeto original.

Una implementación sencilla de SecureTokenManager utiliza un algoritmo de codificación sencillo como, por ejemplo, un algoritmo XOR, para codificar el objeto en un formato serializado y, a continuación, utilizar el algoritmo de decodificación correspondiente para descifrar la señal. Esta implementación no es segura y es fácil quebrantarla.

Implementación predeterminada de WebSphere eXtreme Scale

WebSphere eXtreme Scale proporciona una implementación disponible de forma inmediata para esta interfaz. Esta implementación predeterminada utiliza un par de claves para firmar y verificar la firma y utiliza una clave secreta para cifrar el contenido. Cada servidor tiene un almacén de claves de tipo JCKES donde se almacena el par de claves, una clave privada y una clave pública, y un clave secreta. El almacén de claves tiene que ser de tipo JCKES para poder almacenar las claves secretas. Estas claves se utilizan para cifrar y firmar o verificar la serie secreta en el envío. Además, la señal se asocia con un tiempo de caducidad. En el extremo receptor, los datos se verifican, se descifran y se comparan con la serie secreta del receptor. Los protocolos de comunicación SSL (Secure Sockets Layer) no son obligatorios para la autenticación entre un par de servidores porque las claves privadas y públicas sirven para ese mismo propósito. No obstante, si la comunicación del servidor no está cifrada, los datos podrían robarse con sólo observar la comunicación. Como la señal caduca pronto, la amenaza de ataque de reproducción se minimiza. Esta posibilidad disminuye en gran medida si todos los servidores se despliegan detrás de un cortafuegos.

La desventaja de este enfoque es que los administradores de WebSphere eXtreme Scale deben generar claves y transportarlas a todos los servidores, que pueden provocar una violación de seguridad durante el transporte.

Seguridad JMX (Java Management Extensions)

Puede proteger las invocaciones de beans gestionados (MBean) en un entorno de despliegue dinámico.

Para obtener más información sobre los MBeans disponibles, consulte el apartado “Utilización de beans gestionados (MBeans) para administrar el entorno” en la página 286.

En la topología de despliegue dinámico, los MBeans se alojan directamente en los servidores de catálogo y servidores de contenedor. En general, la seguridad JMX de la topología de despliegue dinámico cumple la especificación de la seguridad JMX como se especifica en la especificación JMX (Java Management Extensions). Consta de las tres partes siguientes:

1. Autenticación: el cliente remoto debe autenticarse en el servidor conector.
2. Control de accesos: el control de accesos de MBean limita quién puede acceder a la información de MBean y quién puede realizar las operaciones de MBean.
3. Transporte seguro: el transporte entre el cliente y el servidor JMX se puede proteger utilizando TLS/SSL.

Autenticación

JMX proporciona métodos para que los servidores de tipo conector autentiquen los clientes remotos. Para el conector RMI, la autenticación se realiza al suministrar un objeto que implemente la interfaz JMXAuthenticator al crear el servidor conector. Así, eXtreme Scale implementa esta interfaz JMXAuthenticator para utilizar el

plug-in ObjectGrid Authenticator para autenticar los clientes remotos. Consulte la guía de aprendizaje de seguridad en *Visión general del producto* para ver los detalles sobre cómo eXtreme Scale autentica un cliente.

El cliente JMX sigue las API de JMX para proporcionar credenciales y establecer conexión con el servidor conector. La infraestructura JMX pasa la credencial al servidor conector, y después llama a la implementación de JMXAuthenticator para obtener la autenticación. Como se ha descrito anteriormente, la implementación de JMXAuthenticator delega la autenticación a la implementación de ObjectGrid Authenticator.

Observe el ejemplo siguiente que describe cómo establecer conexión con un servidor conector mediante una credencial:

```
javax.management.remote.JMXServiceURL jmxUrl = new JMXServiceURL(
        "service:jmx:rmi:///jndi/rmi://localhost:1099/objectgrid/MBeanServer");

    environment.put(JMXConnector.CREDENTIALS, new UserPasswordCredential("admin", "xxxxxx"));

    // Crear JMXCconnectorServer
    JMXConnector cntor = JMXConnectorFactory.newJMXConnector(jmxUrl, null);

    // Conectar e invocar una operación en MBeanServer remoto
    cntor.connect(environment);
```

En este ejemplo, se proporciona una credencial UserPasswordCredential con el ID de usuario establecido en admin y la contraseña establecida en xxxxx. Este objeto UserPasswordCredential se establece en la correlación de entorno, que se utiliza en el método JMXConnector.connect(Map). Este objeto UserPasswordCredential se pasa al servidor a través de la infraestructura de JMX y, finalmente, se pasa a la infraestructura de autenticación de ObjectGrid para la autenticación.

El modelo de programación de cliente cumple la especificación JMX de manera estricta.

Control de acceso

Un servidor MBean JMX puede tener acceso a información confidencial y realizar operaciones confidenciales. JMX ofrece el control de acceso necesario que identifica qué clientes pueden acceder a la información y qué clientes pueden llevar a cabo las operaciones. El control de accesos se crea en el modelo de seguridad Java estándar definiendo los permisos que controlan el acceso al servidor MBean y sus operaciones.

Para el control de accesos o la autorización de la operación JMX, eXtreme Scale se basa en el soporte JAAS proporcionado por la implementación de JMX. En un punto dado de la ejecución de un programa, hay un conjunto de permisos actuales que una hebra de ejecución contiene. Cuando dicha hebra llama a una operación de la especificación JMX, estos permisos se denominan permisos mantenidos. Cuando se realiza una operación JMX, se realiza una comprobación de seguridad para verificar que el permiso necesario está implicado en el permiso mantenido.

La definición de política MBean sigue el formato de la política Java. Por ejemplo, la política siguiente otorga a todos los firmantes y a todas las bases de código el derecho a recuperar la dirección JMX del servidor para PlacementServiceMBean, pero con una restricción para el dominio com.ibm.websphere.objectgrid.

```
grant {
    permission javax.management.MBeanPermission
        "com.ibm.websphere.objectgrid.management.PlacementServiceMBean#retrieveServerJMXAddress [com.ibm.websphere.objectgrid:*,type=PlacementService]",
"invoke";
}
```

Puede utilizar el siguiente ejemplo de política para completar la autorización basada en la identidad de cliente remoto. La política otorga al mismo permiso MBean como se muestra en el ejemplo anterior, excepto que sólo para los usuarios con el nombre X500Principal como 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";
}
```

Las políticas Java sólo se comprueban si el gestor de seguridad está activo. Inicie los servidores de catálogo y los servidores de contenedor con el argumento JVM -Djava.security.manager para aplicar el control de acceso de operaciones MBean.

Transporte seguro

El transporte entre el cliente JMX y el servidor puede protegerse mediante TLS/SSL. Si el valor transportType del servidor de catálogo o servidor de contenedor está establecido en SSL_Required o SSL_Supported, utilice SSL para conectarse al servidor JMX.

Para utilizar SSL, debe configurar el almacén de confianza, el tipo y la contraseña del almacén de confianza en el cliente MBean mediante el uso de las propiedades del sistema -D:

1. -Djavax.net.ssl.trustStore=TRUST_STORE_LOCATION
2. -Djavax.net.ssl.trustStorePassword=TRUST_STORE_PASSWORD
3. -Djavax.net.ssl.trustStoreType=TRUST_STORE_TYPE

Si utiliza com.ibm.websphere.ssl.protocol.SSLSocketFactory como fábrica de socket SSL en el archivo JAVA_HOME/jre/lib/security/java.security, utilice las propiedades siguientes:

1. -Dcom.ibm.ssl.trustStore=TRUST_STORE_LOCATION
2. -Dcom.ibm.ssl.trustStorePassword=TRUST_STORE_PASSWORD
3. -Dcom.ibm.ssl.trustStoreType=TRUST_STORE_TYPE

Archivo XML de descriptor de seguridad

Utilice un archivo XML de descriptor de seguridad de ObjectGrid para configurar una topología de despliegue de eXtreme Scale con la seguridad habilitada. Este tema proporciona archivos XML de ejemplo para ilustrar distintas configuraciones.

Cada elemento y atributo del archivo XML del clúster se describe en la siguiente lista. Utilice los ejemplos para aprender cómo utilizar estos elementos y atributo para configurar el entorno.

Elemento securityConfig

El elemento securityConfig es el elemento de nivel superior del archivo XML de seguridad de ObjectGrid. Este elemento configura el espacio de nombres del archivo y la ubicación del esquema. El esquema se define en el archivo `objectGridSecurity.xsd`.

- Número de apariciones: una
- Elementos hijo: security

Elemento security

Utilice el elemento security para definir una seguridad de ObjectGrid.

- Número de apariciones: una
- Elementos hijo: authenticator, adminAuthorization y systemCredentialGenerator

Atributos

securityEnabled
Habilita la seguridad para la cuadrícula cuando se establece en true. El valor predeterminado es false. Si el valor está establecido en false, la seguridad de nivel de cuadrícula está inhabilitada. Para obtener más información, consulte "Seguridad de la cuadrícula" en la página 310. (Opcional)

singleSignOnEnabled
Permite a un cliente conectarse a cualquier servidor una vez que se ha autenticado con uno de los servidores, si el valor está establecido en true. De lo contrario, un cliente debe autenticarse con cada servidor antes de que se pueda conectar el cliente. El valor predeterminado es false. (Opcional)

loginSessionExpirationTime
Especifica la cantidad de tiempo en segundos antes de que caduque el inicio de sesión. Si la sección de inicio de sesión caduca, el cliente debe volver a autenticarse. (Opcional)

adminAuthorizationEnabled
Habilita la autorización administrativa. Si el valor está establecido en true, todas las tareas administrativas necesitan autorización. El mecanismo de autorización que se utiliza se especifica mediante el valor del atributo adminAuthorizationMechanism. El valor predeterminado es false. (Opcional)

adminAuthorizationMechanism
Indica qué mecanismo de autorización utilizar. WebSphere eXtreme Scale soporta dos mecanismos de autorización: la autorización JAAS (Java Authentication and Authorization Service) y la autorización personalizada. El mecanismo de autorización JAAS utiliza el enfoque basado en la política JAAS estándar. Para especificar JAAS como mecanismo de autorización, establezca el valor en AUTHORIZATION_MECHANISM_JAAS. El mecanismo de autorización personalizada utiliza una implementación de AdminAuthorization de plug-in de usuario. Para especificar un mecanismo de autorización personalizada, establezca el valor en AUTHORIZATION_MECHANISM_CUSTOM. Para obtener más información sobre cómo se utilizan estos dos mecanismos, consulte "Autorización de cliente de aplicaciones" en la página 314. (Opcional)

El siguiente archivo `security.xml` es una configuración de ejemplo para habilitar la seguridad de la cuadrícula de eXtreme Scale.

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>
```

Elemento authenticator

Autentica los clientes en los servidores eXtreme Scale de la cuadrícula. La clase que se especifica mediante el atributo className debe implementar la interfaz com.ibm.websphere.objectgrid.security.plugins.Authenticator. El autenticador puede utilizar las propiedades para llamar a métodos en la clase que se especifica mediante el atributo className. Consulte el elemento property para obtener más información sobre la utilización de propiedades.

En el archivo de ejemplo `security.xml` anterior, la clase com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator se especifica como el autenticador. Esta clase implementa la interfaz com.ibm.websphere.objectgrid.security.plugins.Authenticator.

- Número de apariciones: cero o ninguna
- Elemento hijo: property

Atributos

className
Especifica una clase que implementa la interfaz com.ibm.websphere.objectgrid.security.plugins.Authenticator. Utilice esta clase para autenticar los clientes en los servidores de la cuadrícula de eXtreme Scale. (Necesario)

Elemento adminAuthorization

Utilice el elemento adminAuthorization para configurar el acceso administrativo a la cuadrícula.

- Número de apariciones: cero o ninguna
- Elemento hijo: property

Atributos

className
Especifica una clase que implementa la interfaz com.ibm.websphere.objectgrid.security.plugins.AdminAuthorization. (Necesario)

Elemento systemCredentialGenerator

Utilice un elemento systemCredentialGenerator para configurar un generador de credenciales del sistema. Este elemento sólo se aplica a un entorno dinámico. En el modelo de configuración dinámica, el servidor de contenedor dinámico se conecta al servidor de catálogo como un cliente de eXtreme Scale y el servidor de catálogo se puede conectar al servidor de contenedor de eXtreme Scale también como un cliente. Este generador de credenciales del sistema se utiliza para representar una fábrica de la credencial del sistema.

- Número de apariciones: cero o ninguna
- Elemento hijo: property

Atributos

className
Especifica una clase que implementa la interfaz com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. (Necesario)

Consulte el archivo `security.xml` anterior para ver un ejemplo sobre cómo utilizar systemCredentialGenerator. En este ejemplo, el generador de credenciales del sistema es un com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenCredentialGenerator, que recupera el objeto RunAs Subject de la hebra.

Elemento property

Llama a los métodos set en las clases authenticator y adminAuthorization. El nombre de la propiedad corresponde al método set del atributo className del elemento authenticator o adminAuthorization.

- Número de apariciones: cero o más
- Elemento hijo: property

Atributos

name
Especifica el nombre de la propiedad. El valor que se asigna a este atributo debe corresponder a un método set de la clase que se proporciona como el atributo className de bean que lo contiene. Por ejemplo, si el atributo className del bean se establece en com.ibm.MyPlugin, y el nombre de la propiedad suministrada es size, la clase com.ibm.MyPlugin debe tener un método setSize. (Necesario)

type
Especifica el tipo de la propiedad. El tipo del parámetro se pasa al método set identificado por el atributo name. Los valores válidos son los primitivos Java, sus correspondientes java.lang y java.lang.String. Los atributos de nombre y tipo deben corresponder a una signatura de método del atributo className del bean. Por ejemplo, si el nombre es size y el tipo es int, entonces debe existir un método setSize(int) en la clase que se especifica como el atributo className para el bean. (Necesario)

value
Especifica el valor de la propiedad. Este valor se convierte en el tipo que se especifica mediante el atributo de tipo y se utiliza como un parámetro en la llamada al método set que se identifica mediante los atributos de nombre y tipo. El valor de este atributo no se valida de ningún modo. El implementador del plug-in debe verificar que el valor que se ha pasado es válido. (Necesario)

description
Proporciona una descripción de la propiedad. (Opcional)

Si desea más información, consulte "Archivo `objectGridSecurity.xsd`" en la página 195.

Integración de la seguridad con WebSphere Application Server

WebSphere eXtreme Scale proporciona varias características de seguridad para integrarse con la infraestructura de seguridad de WebSphere Application Server.

Integración de autenticación

Cuando los clientes y los servidores eXtreme Scale se ejecutan en WebSphere Application Server y en el mismo dominio de seguridad, puede utilizar la infraestructura de seguridad de WebSphere Application Server para propagar las credenciales de autenticación de cliente en el servidor eXtreme Scale. Por ejemplo, si un servlet actúa como un cliente de eXtreme Scale para conectarse a un servidor eXtreme Scale en el mismo dominio de seguridad, y el servlet ya ha sido autenticado, es posible propagar la señal de autenticación del cliente (servlet) al servidor y, a continuación, utilice la infraestructura de seguridad de WebSphere Application Server para volver a convertir la señal de autenticación a las credenciales de cliente.

Integración de seguridad distribuida con WebSphere Application Server

Para el modelo ObjectGrid distribuido, la integración de seguridad puede completarse utilizando las siguientes clases:

```
com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator
com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator
com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredential
```

Si desea más información, consulte “Autenticación de cliente de aplicaciones” en la página 311.En el siguiente ejemplo se muestra cómo utilizar la clase WSTokenCredentialGenerator:

```
/**
 * conectarse al servidor ObjectGrid.
 */
protected ClientClusterContext connect() throws ConnectException {
  ClientSecurityConfiguration csConfig = ClientSecurityConfigurationFactory
  .getClientSecurityConfiguration(proFile);

  CredentialGenerator gen = getWSCredGen();

  csConfig.setCredentialGenerator(gen);

  return objectGridManager.connect(csConfig, null);
}

/**
 * Obtener un WSTokenCredentialGenerator
 *
private CredentialGenerator getWSCredGen() {
    WSTokenCredentialGenerator gen = new WSTokenCredentialGenerator(
        WSTokenCredentialGenerator.RUN_AS_SUBJECT);
    return gen;
}
```

En el lado del servidor, utilice el autenticador WSTokenAuthentication para autenticar el objeto WSTokenCredential.

Integración de seguridad local con WebSphere Application Server

Para el modelo de ObjectGrid local, la integración de seguridad se puede realizar utilizando las dos clases siguientes:

- com.ibm.websphere.objectgrid.security.plugins.builtins.WSSubjectSourceImpl
- com.ibm.websphere.objectgrid.security.plugins.builtins.WSSubjectValidationImpl

Si desea más información sobre estas clases, consulte la información sobre la seguridad local en *Guía de programación*. Puede configurar la clase WSSubjectSourceImpl como el plug-in SubjectSource y la clase WSSubjectValidationImpl como el plug-in SubjectValidation.

Inicio y parada de servidores eXtreme Scale seguros

A menudo, los servidores necesitan ser seguros para el entorno de despliegue, que requiere una configuración específica.

Inicio de un servidor seguro en un entorno Java SE

Puede iniciar un servicio de catálogo o servidores de contenedor, del modo siguiente.

Inicio de un servicio de catálogo eXtreme Scale seguro

El inicio de un proceso de servicio de catálogo eXtreme Scale seguro requiere dos archivos de configuración de seguridad más:

Archivo XML de descriptor de seguridad: el archivo XML de descriptor de seguridad describe las propiedades de seguridad comunes a todos los servidores (incluidos los servidores de catálogo y los servidores de contenedor). Un ejemplo de propiedad es la configuración de autenticador que representa el mecanismo de autenticación y el registro de usuarios.

Archivo de propiedades de servidor. El archivo de propiedades de servidor configura las propiedades de seguridad específicas del servidor.

Cuando se utiliza el mandato startOgServer.sh o startOgServer.cat para iniciar un proceso de servicio de catálogos eXtreme Scale seguro, puede utilizar -clusterSecurityFile o -clusterSecurityUrl para establecer el archivo XML de descriptor de seguridad como un tipo de archivo o un tipo de URL y puede utilizar –serverProps para establecer el archivo de propiedades de servidor.

Inicio de un servidor de contenedor eXtreme Scale seguro

El inicio de un servidor de contenedor eXtreme Scale seguro requiere un archivo de configuración de seguridad:

- **Archivo de propiedad de servidor:** el archivo de propiedad de servidor configura las propiedades de seguridad específicas del servidor. Consulte “Archivo de propiedades de servidor” en la página 197 si desea más detalles.

Cuando se utiliza el mandato startOgServer.sh o startOgServer.cat para iniciar un servidor de contenedor eXtreme Scale seguro, pueda utilizar –serverProps para establecer el archivo de propiedades de servidor. Existen más métodos para establecer el archivo de propiedad de servidor, consulte el archivo de propiedades de servidor, si desea más detalles.

Si desea más detalles sobre cómo utilizar el mandato startOgServer.sh o startOgServer.bat y sus opciones, consulte “Script startOgServer” en la página 235.

Parada de un servidor eXtreme Scale seguro

La parada de un proceso de servicio de catálogo de eXtreme Scale seguro o un servidor de contenedor requiere un archivo de configuración de seguridad:

- **archivo de propiedades de cliente** el archivo de propiedades de cliente se puede utilizar para configurar las propiedades de servidor del cliente. Las propiedades

de seguridad de cliente son necesarias para que un cliente se conecte a un servidor seguro. Consulte “Archivo de propiedades de cliente” en la página 203 si desea más detalles.

Cuando se utiliza el mandato stopOgServer.sh o stopOgServer.cat para detener un proceso de servicio de catálogos de eXtreme Scale seguro o un servidor de contenedor, puede utilizar -clientSecurityFile para establecer las propiedades de seguridad de cliente.

Si desea más detalles sobre cómo utilizar el mandato stopOgServer.sh o stopOgServer.cat y sus opciones, consulte “Script stopOgServer” en la página 241.

Inicio de un servidor seguro en WebSphere Application Server

El inicio de un servidor ObjectGrid seguro en WebSphere Application Server es similar al inicio de un servidor ObjectGrid no seguro, excepto que debe pasar los archivos de configuración de seguridad. En lugar de utilizar -[PROPERTY_FILE] (por ejemplo -serverProps) en el mandato como en el entorno Java SE, utilice -D[PROPERTY_FILE] en los argumentos genéricos de la máquina virtual Java (JVM).

Inicio de un servicio de catálogo seguro en Websphere Application Server

Un servidor de catálogo contiene dos niveles diferentes de información de seguridad:

- `-Dobjectgrid.cluster.security.xml.url`: esto especifica el archivo objectGridSecurity.xml que describe las propiedades de seguridad comunes a todos los servidores (incluidos los servidores de catálogo y servidores de contenedor). Un ejemplo es la configuración del autenticador que representa el registro de usuarios y el mecanismo de autenticación. El nombre de archivo especificado para esta propiedad debe tener un formato de URL como, por ejemplo, ″file:///tmp/og/objectGridSecurity.xml″.
- `-Dobjectgrid.server.props`: esto especifica el archivo de propiedades del servidor que contiene las propiedades de seguridad específicas del servidor. El nombre de archivo especificado para esta propiedad tiene un formato de vía de acceso de archivo plano como, por ejemplo, ″c:/tmp/og/catalogserver.props″. Tenga en cuenta que el uso de -Dobjectgrid.security.server.props está en desuso, pero puede seguir utilizándolo para la compatibilidad con versiones anteriores.

Para iniciar un servicio de catálogos seguro en WebSphere Application Server, siga el apartado ″Incorporado en WebSphere Application Server″ en “Seguridad de la cuadrícula” en la página 310.

A continuación, establezca la propiedad de seguridad en el argumento genérico de JVM del proceso.

```
-Dobjectgrid.cluster.security.xml.url=file:///tmp/og/
objectGridSecurity.xml-Dobjectgrid.server.props=/tmp/og/
catalog.server.props
```

Los pasos para añadir los argumentos genéricos de JVM son los siguientes:

- Expanda ″Administración del sistema″ en la vista de tarea de la izquierda.
- Pulse el proceso de WebSphere Application Server en el que se despliega el servicio de catálogos, por ejemplo, el ″Gestor de despliegue″.

- En la página de la derecha, expanda ″Java y gestión de procesos″ bajo ″Infraestructura de servidor″.
- Pulse ″Definición de proceso″.
- Pulse ″Máquina virtual Java″ bajo ″Propiedades adicionales″.
- Escriba las propiedades en el recuadro de texto de argumentos de JVM genéricos.

Inicio de un servidor de contenedor seguro en WebSphere Application Server

Un servidor de contenedor, cuando se conecta al servidor de catálogo, obtendrá todas las configuraciones de seguridad configuradas en objectGridSecurity.xml como, por ejemplo, el valor de configuración del autenticador o el de tiempo de espera de inicio se sesión. Asimismo, un servidor de contenedor debe configurar sus propias propiedades de seguridad específicas del servidor en la propiedad -Dobjectgrid.server.props.

Debe utilizar la propiedad -Dobjectgrid.server.props, en lugar de la propiedad -Dobjectgrid.security.server.propsproperty, porque también se colocan otras propiedades relacionadas sin seguridad en este archivo de propiedades. El nombre de archivo especificado para esta propiedad está en formato de vía de acceso de archivo sencillo como, por ejemplo, `c:/tmp/og/server.props`.

Siga los mismos pasos anteriores para añadir la propiedad de seguridad a los argumentos genéricos de la JVM.

Tareas relacionadas

“Inicio de servidores de WebSphere eXtreme Scale autónomos” en la página 228
Cuando se ejecuta una configuración de WebSphere eXtreme Scale autónoma, el entorno está formado por servidores de catálogo, servidores de contenedor y procesos de cliente de eXtreme Scale. Debe configurar e iniciar manualmente estos procesos.

“Inicio del servicio de catálogo en un entorno autónomo” en la página 229
Debe iniciar el servicio de catálogo manualmente cuando utilice un entorno distribuido de WebSphere eXtreme Scale que no ejecute WebSphere Application Server.

“Inicio de procesos de contenedor” en la página 232
Puede iniciar eXtreme Scale desde la línea de mandatos, utilizando una topología de despliegue, o utilizando un archivo `server.properties`.

Referencia relacionada

“Script startOgServer” en la página 235
El script startOgServer inicia servidores. Puede utilizar diversos parámetros al iniciar los servidores para habilitar el rastreo, especificar números de puerto, etc.

“Registros y rastreo” en la página 303
Puede utilizar los registros y el rastreo para supervisar y resolver problemas del entorno. Los registros están en distintas ubicaciones en función de su configuración. Es posible que sea necesario proporcionar el rastreo para un servidor cuando se trabaja con el servicio de soporte IBM.

Capítulo 10. Resolución de problemas

Además de los registros y el rastreo, los mensajes y las notas del release, puede utilizar las herramientas de supervisión para solucionar los problemas de la configuración.

Utilización de herramientas de supervisión para la resolución de problemas

Además de los registros y el rastreo, los mensajes y las notas de release, puede utilizar herramientas de supervisión para descubrir cuestiones como, por ejemplo, la ubicación de los datos en el entorno, la disponibilidad de los servidores en la cuadrícula, etc. Si está trabajando en un entorno WebSphere Application Server, podrá utilizar la infraestructura PMI (Performance Monitoring Infrastructure). Si está trabajando en un entorno autónomo, podrá utilizar una herramienta de supervisión de proveedor como, por ejemplo, CA Wily Introscope o Hyperic HQ. También puede utilizar y personalizar el programa de utilidad xsAdmin de ejemplo para mostrar la información textual sobre el entorno.

Si desea más información sobre las herramientas de supervisión, consulte Capítulo 8, “Supervisión del entorno de despliegue”, en la página 261.

Registros y rastreo

Puede utilizar los registros y el rastreo para supervisar y resolver problemas del entorno. Los registros están en distintas ubicaciones en función de su configuración. Es posible que sea necesario proporcionar el rastreo para un servidor cuando se trabaja con el servicio de soporte IBM.

Registros con WebSphere Application Server

Consulte WebSphere Application Server Information Center para obtener más información.

Registros con WebSphere eXtreme Scale en un entorno autónomo

Con servidores de catálogo autónomo y de contenedor, establezca la ubicación de los registros y toda especificación de rastreo. Los registros del servidor de catálogo se encuentran en la ubicación en la que ejecutó el mandato de inicio de servidor.

Establecimiento de la ubicación de registro para los servidores de contenedor

De forma predeterminada, los registros de un contenedor están en el directorio en el que se ha ejecutado el mandato de servidor. Si inicia los servidores en el directorio `<inicio_eXtremeScale>/bin`, los registros y los archivos de rastreo se encuentran en el directorio `logs/<nombre_servidor>` del directorio `bin`. Para especificar una ubicación alternativa de los registros de un servidor de contenedor, cree un archivo de propiedades, como el archivo `server.properties`, con el siguiente contenido:

```
workingDirectory=<directorio>
traceSpec=
systemStreamToFileEnabled=true
```

La propiedad workingDirectory es el directorio raíz para los registros y el archivo de rastreo opcional. WebSphere eXtreme Scale crea un directorio con el nombre del servidor de contenedor con un archivo `SystemOut.log`, un archivo `SystemErr.log` y un archivo de rastreo si el rastreo se ha habilitado con la opción traceSpec. Para utilizar un archivo de propiedades durante el inicio del contenedor, utilice la opción **-serverProps** y proporcione la ubicación del archivo de propiedades de servidor.

Consulte "Inicio de servidores de WebSphere eXtreme Scale autónomos" en la página 228 y "Script startOgServer" en la página 235 para obtener más información.

Los mensajes de información común que se deben buscar en el archivo `SystemOut.log` son mensajes de confirmación de inicio. Si desea más información sobre un mensaje específico, consulte "Mensajes" en la página 335.

Rastreo con WebSphere Application Server

Consulte WebSphere Application Server Information Center para obtener más información.

Rastreo en el servicio de catálogo

Puede establecer el rastreo en un servicio de catálogo utilizando los parámetros **-traceSpec** y **-traceFile** durante el inicio del servicio de catálogo. Por ejemplo:

```
startOgServer.sh catalogServer -traceSpec
ObjectGridPlacement=all=enabled -traceFile
/home/user1/logs/trace.log
```

Si inicia el servicio de catálogo en el directorio <*inicio_eXtremeScale*>/bin, los registros y los archivos de rastreo estarán en un directorio logs/<*nombre_servicio_catálogo*> en el directorio bin. Consulte "Inicio del servicio de catálogo en un entorno autónomo" en la página 229 si desea más detalles sobre cómo iniciar un servicio de catálogo.

Rastreo en un servidor de contenedor autónomo

Puede habilitar el rastreo en un servidor de contenedor de dos formas. Puede crear un archivo de propiedades de servidor tal como se explica en la sección de registros, o puede habilitar el rastreo utilizando la línea de mandatos durante el inicio. Para habilitar el rastreo de contenedor con un archivo de propiedades de servidor, actualice la propiedad **traceSpec** con la especificación de rastreo necesaria. Para habilitar el rastreo de contenedor utilizando parámetros de inicio, utilice los parámetros **-traceSpec** y **-traceFile**. Por ejemplo:

```
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
```

Si inicia el servidor en el directorio <*inicio_eXtremeScale*>/bin, los registros y archivos de rastreo se encuentran en los directorios logs/<*nombre_servidor*> en el directorio bin Consulte "Inicio de procesos de contenedor" en la página 232 si desea más detalles sobre cómo iniciar un proceso de contenedor.

Rastreo con la interfaz ObjectGridManager

Otra opción es establecer el rastreo durante la ejecución de una interfaz ObjectGridManager. Si se establece el rastreo en una interfaz ObjectGridManager,

se puede utilizar para obtener el rastreo en un cliente de eXtreme Scale, mientras se conecta a eXtreme Scale y confirma transacciones. Para establecer el rastreo en una interfaz ObjectGridManager, proporcione una especificación de rastreo y un registro de rastreo.

```
ObjectGridManager manager = ObjectGridManagerFactory.getObjectGridManager();
...
manager.setTraceEnabled(true);
manager.setTraceFileName("logs/myClient.log");
manager.setTraceSpecification("ObjectGridReplication=all=enabled");
```

Habilitación del rastreo con el programa de utilidad xsadmin

Para habilitar el rastreo con el programa de utilidad xsadmin, utilice la opción **setTraceSpec**. Utilice el programa de utilidad xsadmin para habilitar el rastreo en un entorno autónomo durante la ejecución, en lugar de durante el arranque. Puede habilitar el rastreo en todos los servidores y servicios de catálogo o puede filtrar los servidores basándose en el nombre de ObjectGrid, etc. Por ejemplo, para habilitar el rastreo de ObjectGridReplication con acceso al servidor del servicio de catálogo, ejecute:

```
<inicio_eXtremeScale>/bin>xsadmin.bat -setTraceSpec "ObjectGridReplication=all=enabled"
```

También puede inhabilitar el rastreo estableciendo la especificación de rastreo en `*=all=disabled.`

Consulte “Utilización del programa de utilidad de ejemplo xsAdmin” en la página 287 si desea más información.

Archivos y directorio ffdc

Los archivos FFDC sirven para que el servicio de soporte de IBM ayude a realizar la depuración. El servicio de soporte de IBM puede solicitar estos archivos si se produce un problema.

Estos archivos están en un directorio denominado, `ffdc`, y contienen archivos que se parecen al siguiente:

```
server2_exception.log
server2_20802080_07.03.05_10.52.18_0.txt
```

Conceptos relacionados

"Inicio y parada de servidores eXtreme Scale seguros" en la página 327
A menudo, los servidores necesitan ser seguros para el entorno de despliegue, que requiere una configuración específica.

Tareas relacionadas

"Inicio de servidores de WebSphere eXtreme Scale autónomos" en la página 228
Cuando se ejecuta una configuración de WebSphere eXtreme Scale autónoma, el entorno está formado por servidores de catálogo, servidores de contenedor y procesos de cliente de eXtreme Scale. Debe configurar e iniciar manualmente estos procesos.

"Inicio del servicio de catálogo en un entorno autónomo" en la página 229
Debe iniciar el servicio de catálogo manualmente cuando utilice un entorno distribuido de WebSphere eXtreme Scale que no ejecute WebSphere Application Server.

"Inicio de procesos de contenedor" en la página 232
Puede iniciar eXtreme Scale desde la línea de mandatos, utilizando una topología de despliegue, o utilizando un archivo `server.properties`.

Opciones de rastreo

Puede habilitar el rastreo para proporcionar información sobre el entorno al servicio de soporte de IBM.

Sobre el rastreo

El rastreo de WebSphere eXtreme Scale se divide en varios componentes distintos. De forma similar al rastreo de WebSphere Application Server, puede especificar el nivel de rastreo que se debe utilizar. Los niveles comunes de rastreo son: all, debug, entryExit y event.

A continuación se muestra una serie de rastreo de ejemplo:

```
ObjectGridComponent=level=enabled
```

Puede concatenar valores de rastreo. Utilice el símbolo `*` (asterisco) para especificar un valor comodín como, por ejemplo, `ObjectGrid*=all=enabled`. Si necesita proporcionar un rastreo al servicio de soporte de IBM, se solicita una serie de rastreo específica. Por ejemplo, si hay un problema con la réplica, se puede solicitar la serie de rastreo `ObjectGridReplication=debug=enabled`.

Especificación de rastreo

ObjectGrid
Motor de memoria caché principal general.

ObjectGridCatalogServer
Servicio de catálogo general.

ObjectGridChannel
Comunicaciones de topología de despliegue estática.

ObjectgridCORBA
Comunicaciones de topología de despliegue dinámica.

ObjectGridDataGrid
API de AgentManager.

ObjectGridDynaCache
El proveedor de la memoria caché dinámica de WebSphere eXtreme Scale.

ObjectGridEntityManager
La API de EntityManager. Utilícela con la opción Projector.

ObjectGridEvictors
Desalojadores incorporados de ObjectGrid.

ObjectGridJPA
Cargadores JPA (Java Persistence API).

ObjectGridJPACache
Plug-ins de memoria caché JPA.

ObjectGridLocking
Gestor de bloqueos de entradas de memoria caché de ObjectGrid.

ObjectGridMBean
Beans de gestión.

ObjectGridPlacement
Servicio de colocación de fragmentos de servidor de catálogo.

ObjectGridQuery
Consulte de ObjectGrid.

ObjectGridReplication
Servicio de réplica.

ObjectGridRouting
Detalles de direccionamiento de cliente/servidor.

ObjectGridSecurity
Rastreo de seguridad.

ObjectGridStats
Estadísticas de ObjectGrid.

ObjectGridStreamQuery
La API de consulta de secuencia.

ObjectGridWriteBehind
Escritura diferida de ObjectGrid

Projector
El motor dentro de la API de EntityManager.

QueryEngine
El motor de consulta para la API de consulta de objetos y la API de consulta de EntityManager.

QueryEnginePlan
Diagnósticos del plan de consulta.

Mensajes

Cuando encuentre un mensaje en un registro u otras partes de la interfaz del producto, puede buscar el mensaje por su prefijo de componente para descubrir más información.

Cómo encontrar mensajes

Cuando encuentre un mensaje en un registro, copie el número de mensaje con su prefijo de letra y el número y búsquelo en el centro de información (por ejemplo, `CWOBJ1526I`). Cuando busque el mensaje, podrá encontrar una explicación adicional

del mensaje y las posibles acciones que puede llevar a cabo para resolver el problema.

En Information Center encontrará un índice de los mensajes del producto.

Notas del release

Se proporcionan enlaces al sitio web de soporte del producto, a la documentación del producto y a las últimas actualizaciones, limitaciones y problemas conocidos.

- "Acceso a las actualizaciones, limitaciones y problemas conocidos más recientemente"
- "Acceso a los requisitos de software y del sistema"
- "Acceso a documentación del producto"
- "Acceso al sitio web de soporte de producto"
- "Cómo ponerse en contacto con el servicio de soporte de software de IBM"

Acceso a las actualizaciones, limitaciones y problemas conocidos más recientemente

Las notas de release están disponibles en el sitio de soporte del producto como notas técnicas. Para ver una lista de todas las notas técnicas para WebSphere eXtreme Scale, vaya a la página web de soporte.

- Para ver una lista de las notas del release para la versión 7.0, visite la página web de soporte.
- Para ver una lista de las notas del release para la versión 6.1, vaya a la página wiki de las notas del release.

Acceso a los requisitos de software y del sistema

Los requisitos de hardware y software aparecen documentados en las páginas siguientes:

- Requisitos de sistema detallados

Acceso a documentación del producto

Para obtener todo el conjunto de información, vaya a la página de la biblioteca.

Acceso al sitio web de soporte de producto

Para localizar las últimas notas técnicas, descargas, arreglos y otra información relacionada con el soporte, vaya a la página de soporte técnico.

Cómo ponerse en contacto con el servicio de soporte de software de IBM

Si encuentra un problema con el producto, primero intente llevar a cabo las siguientes acciones:

- Siga los pasos descritos en la documentación del producto
- Busque la documentación relacionada en la ayuda en línea
- Busque los mensajes de error en la consulta de mensajes

Si no puede resolver el problema con ninguno de los métodos citados anteriormente, póngase en contacto con el servicio de IBM.

Rendimiento y procedimientos recomendados

Puede mejorar el rendimiento de WebSphere eXtreme Scale.

WebSphere eXtreme Scale ha sido diseñado para mejorar el rendimiento de las aplicaciones. Existen numerosos factores que afectan el rendimiento de la aplicación cuando se utiliza eXtreme Scale. Algunos son el tamaño de la transacción, la modalidad de copia, la técnica de serialización y la arquitectura de particiones, por ejemplo. Para conseguir un rendimiento óptimo, debe definir la arquitectura e implementar la aplicación con detenimiento.

Antes de intentar mejorar el rendimiento, consulte la información sobre las transacciones en *Visión general del producto* si desea más información sobre cómo elegir una estrategia de bloqueo para el entorno.

Rendimiento de artefactos eXtreme Scale

Puede trabajar con diferentes aspectos de eXtreme Scale para mejorar el rendimiento.

- **Rendimiento de la serialización:** Consulte la información sobre el rendimiento de serialización en *Visión general del producto* si desea más información.
- **Tamaño de la transacción:** Consulte la información sobre los tamaños de transacción en *Visión general del producto* si desea más información.
- **Rendimiento de la réplica:** Consulte la información sobre el rendimiento de la réplica en *Visión general del producto* si desea más información.
- **Impacto del rendimiento del gestor de entidades:** consulte la información sobre el rendimiento del gestor de entidades sobre el rendimiento del gestor de entidades en *Guía de programación* si desea más información.
- **Ajuste de consulta:** Consulte la información sobre cómo ajustar las consultar para el rendimiento en *Guía de programación* si desea más información.

Procedimientos recomendados

Puede mejorar el rendimiento de las correlaciones si pone en práctica algunos procedimientos recomendados. Cada aplicación y entorno utiliza una solución distinta para el rendimiento, y eXtreme Scale proporciona personalizaciones incorporadas para mejorar el rendimiento. Pero todavía puede mejorar el rendimiento dentro de la arquitectura de la aplicación. Las mejoras se implementan solo en el contexto de la aplicación y su arquitectura.

- **Procedimientos recomendados para el rendimiento de bloqueo:** elija entre las distintas estrategias de bloqueo que afectan al rendimiento de las aplicaciones. Consulte la información sobre los procedimientos recomendados para el rendimiento de bloqueo en *Guía de programación* si desea más información.
- **Procedimientos recomendados para el método CopyMode:** elija entre las distintas modalidades de copia que se utilizan para cambiar cómo eXtreme Scale mantiene y copia entradas. Consulte la información sobre los procedimientos recomendados para el método CopyMode en *Guía de programación* si desea más información.
- **Procedimientos recomendados para la interfaz ObjectTransformer:** utilice la interfaz ObjectTransformer para permitir devoluciones de llamada a la aplicación que proporcionan implementaciones personalizadas de operaciones comunes y caras como, por ejemplo, la serialización de objetos y la copia exacta de un objeto. Consulte la información sobre los procedimientos recomendados de la interfaz ObjectTransformer en *Guía de programación* si desea más información.

- **Procedimientos recomendados para el rendimiento del desalojador del plug-in:** elija entre las estrategias para los desalojadores utilizados con menor frecuencia (LFU) y los desalojadores menos utilizados recientemente (LRU). Consulte la información sobre los procedimientos recomendados para el desalojador de plug-in en *Guía de programación* si desea más información.
- **Procedimientos recomendados para el desalojador predeterminado:** las propiedades para el desalojador de tiempo de vida (TTL) predeterminado, que es el desalojador predeterminado creado con cada backingMap. Consulte la información sobre los procedimientos recomendados para el desalojador predeterminado en *Guía de programación* si desea más información.
- Ajuste de la JVM
- "Utilización de WebSphere Real Time" en la página 222: esta característica permite una recogida de basura más predecible junto con un tiempo de respuesta estable y coherente y un rendimiento de transacciones con eXtreme Scale.

Supervisión del rendimiento

eXtreme Scale también proporciona un mecanismo de supervisión del rendimiento. Consulte los apartados siguientes para obtener información sobre las herramientas y estadísticas de la supervisión del rendimiento.

- "Módulos PMI" en la página 275
- "Supervisión del rendimiento con PMI de WebSphere Application Server" en la página 270

Capítulo 11. Glosario

Este glosario incluye los términos y las definiciones para WebSphere eXtreme Scale.

En este glosario se utilizan las siguientes referencias cruzadas:

1. Véase remite al lector de un término a una sinónimo preferido, o de un acrónimo o abreviatura a la forma completa definida.
2. Véase también remite al lector a un término relacionado o en contraste.

Para consultar los glosarios de otros productos IBM, vaya a www.ibm.com/software/globalization/terminology.

Acuerdo de nivel de servicio (SLA) . Un contrato entre un cliente y un proveedor de servicios que especifica las expectativas para el nivel de servicio con respecto a la disponibilidad, el rendimiento y otros objetivos mensurables.

administrador. La persona responsable de las tareas administrativas tales como la autorización de accesos y la gestión de contenidos. Los administradores pueden también otorgar los niveles de autoridad a los usuarios.

administrador de seguridad. La persona que controla el acceso a los datos de la empresa y a las funciones de programa.

afinidad de sesiones. Un método de configurar las aplicaciones de modo que un cliente siempre está conectado al mismo servidor. Estas configuraciones inhabilitan la gestión de cargas de trabajo después de la conexión inicial obligando a una solicitud de cliente a ir siempre al mismo servidor.

agente. Programa que realiza una acción en nombre de un usuario u otro programa sin la intervención del usuario o en una planificación regular, e informa de los resultados al usuario o programa.

agente de nodo. Un agente administrativo que gestiona todos los servidores de aplicaciones de un nodo y que representa el nodo de la célula de gestión.

alias de autenticación. Un alias que autoriza el acceso a los adaptadores de recursos y los orígenes de datos. Un alias de autenticación contiene datos de autenticación, como ID de usuario y contraseña.

almacén de certificados de colecciones. Una colección de certificados intermedios o listas de revocación de certificados, CRL, que utilizan una vía de acceso a certificados para crear una cadena de certificados para su validación.

almacén de datos persistente. Almacenamiento no volátil para datos de sucesos, como un sistema de base de datos, que se mantiene a través de los límites de sesión y que continúa existiendo después de la ejecución del programa o proceso que la ha creado.

alta disponibilidad (HA) . Que pertenece a un sistema en clúster que se ha vuelto a configurar cuando se producen anomalías de nodo o daemon, de forma que las cargas de trabajo se pueden redistribuir a los nodos restantes del clúster.

ámbito.

1. Una especificación del límite dentro del que se pueden utilizar los recursos del sistema.

2. En los servicios web, una propiedad que identifica el ciclo de vida del objeto que presta servicio a la solicitud de invocación.

analista de sistemas. Un especialista responsable de traducir requisitos de negocio en definiciones y soluciones del sistema.

APAR. Véase informe autorizado de análisis de programa.

API. Véase interfaz de programación de aplicaciones.

API de JavaMail. Una infraestructura independiente de la plataforma y del protocolo que permite crear aplicaciones cliente de correo basadas en Java.

API de Java para XML (JAX) . Un conjunto de API basadas en Java para manejar distintas operaciones que impliquen datos definidos mediante XML (lenguaje de códigos ampliable.)

aplicación. Uno o más programas de sistema o componentes de software que proporcionan una función de soporte directo de un proceso o procesos de negocio específicos.

aplicación cliente. Una aplicación, que se ejecuta en una estación de trabajo y está enlazada con un cliente, que da acceso a la aplicación para poner servicios en cola en un servidor.

aplicación Java EE. Cualquier unidad desplegable de funcionalidad Java EE. Esta unidad puede ser un único módulo o un grupo de módulos empaquetados en un archivo EAR con un descriptor de despliegue de aplicación Java EE. (Sun)

archivador empresarial (EAR) . Tipo especializado de archivo JAR, definido por el estándar Java EE, utilizado para desplegar aplicaciones Java EE en servidores de aplicaciones Java EE. Un archivo EAR contiene componentes EJB, un descriptor de despliegue y archivos WAR (archivador web) para aplicaciones web individuales. Véase también archivador web.

archivador Java. Formato de archivo comprimido para almacenar todos los recursos necesarios para instalar y ejecutar un programa Java en un solo archivo. Véase también archivador web, archivador empresarial.

archivo de almacén de confianza . Archivo de base de datos de claves que contiene las claves públicas de una entidad de confianza.

archivo de clase. Un archivo fuente Java compilado.

archivo de definición de build. Archivo XML que identifica los componentes y características de un paquete de instalación personalizado (CIP).

archivo de exportación.

1. Un archivo creado durante el proceso de desarrollo para las operaciones de entrada que contienen los valores de configuración para el proceso entrante.

2. El archivo que contiene los datos que se han exportado.

archivo delimitado por comas. Un archivo cuyos registros contienen campos que están separados por una coma.

archivo JAR. Un archivo Java de archivado. Véase también archivador web, archivador empresarial.

archivo JAR EJB. Archivador Java que contiene un módulo EJB. (Sun)

archivo Java. Archivo fuente editable (con la extensión .java) que se puede compilar en bytecode (archivo .class).

archivo JSP. Archivo HTML de script con una extensión .jsp que permite la inclusión de contenido dinámico en páginas web. Un archivo JSP se puede solicitar directamente como un URL, llamado por un servlet o desde una página HTML.

área del editor. En Eclipse área de la ventana del entorno de trabajo donde se abren los archivos para editarlos.

arreglo temporal. Un arreglo certificado que está generalmente disponible para todos los clientes entre paquetes de arreglos planificados regularmente o entre releases. Véase también fixpack.

Arreglo temporal de programa (PTF) . Para los productos de System i, System p y System z, un arreglo probado por IBM que se pone a disposición de todos los clientes. Véase también fixpack.

asíncrono. Relativo a sucesos que no están sincronizados en el tiempo o que no se producen a intervalos de tiempo regulares o predecibles.

atributo global. En XML, un atributo declarado como hijo del elemento de esquema en lugar de como parte de una definición de tipo complejo. Es posible hacer referencia a los atributos globales en uno o varios modelos de contenido utilizando el atributo ref.

autenticación. Un servicio de seguridad que proporciona la prueba de que un usuario de un sistema es realmente quien dice ser. Los mecanismos habituales para implementar este servicio son contraseñas y firmas digitales. La autenticación es distinta de la autorización; la autenticación no se ocupa de garantizar ni de denegar el acceso a los recursos del sistema.

autónomo. Independiente de cualquier otro dispositivo, programa o sistema. En un entorno de red, una máquina autónoma accede localmente a todos los recursos necesarios.

autorización. El proceso para otorgar a un usuario, sistema o proceso, un acceso completo o restringido a un objeto, recurso o función.

base de datos local. Base de datos que está ubicada en la estación de trabajo en uso.

bean. Definición o instancia de un componente JavaBeans. Véase también JavaBeans, enterprise bean.

bean de entidad. En programación EJB, un enterprise bean que representa los datos persistentes mantenidos en una base de datos. Cada bean de entidad transporta su propia identidad. (Sun)

bean de mandato. Un proxy que puede invocar una sola operación utilizando un método execute().

Bean gestionado (MBean) . En la especificación de JMX (Java Management Extensions), los objetos Java que implementan recursos y su instrumentación.

Bean Scripting Framework. Una arquitectura para incorporar funciones de lenguaje de scripts en las aplicaciones Java.

biblioteca.

1. Colección de elementos de modelo, que incluyen elementos de negocio, procesos, tareas, recursos y organizaciones.

2. Proyecto que se utiliza para el desarrollo, gestión de versiones y organización de los recursos compartidos. Sólo se puede crear y almacenar un subconjunto de tipos de artefactos en una biblioteca como, por ejemplo, objetos empresariales e interfaces.

bifurcación. Un elemento de proceso que hace copias de sus entradas y las reenvía en paralelo por varias vías de proceso distintas.

bloqueo. Procedimiento que impide que un proceso de aplicación perciba los cambios no confirmados que realiza otro proceso de aplicación, y que además impide que un proceso de aplicación actualice los datos a los que está accediendo otro proceso. Un bloqueo garantiza la integridad de los datos al impedir el acceso simultáneo de varios usuarios a datos incoherentes.

bloqueo actualizable. Bloqueo que identifica el intento de actualizar una entrada de la memoria caché al utilizar un bloqueo pesimista.

bloqueo compartido. Bloqueo que limita a los procesos de aplicaciones de ejecución simultánea a operaciones de solo lectura de los datos de la base de datos.

bloqueo exclusivo. Bloqueo que impide que los procesos de aplicaciones de ejecución simultánea accedan a los datos de la base de datos. Véase también bloqueo compartido.

bloqueo pesimista. Estrategia de bloqueo según la cual un bloqueo tiene lugar entre el tiempo durante el que se selecciona una fila y el tiempo durante el cual se realiza una operación de actualización o supresión en dicha fila.

BMP. Véase persistencia gestionada por bean.

BMT. Véase transacción gestionada por bean.

bucle. Una secuencia de instrucciones realizada repetidamente.

bucle do while. Bucle que repite la misma secuencia de actividades siempre que se cumpla una condición. A diferencia de un bucle while, un bucle do while comprueba la condición al final del bucle. Eso significa que la secuencia de actividades siempre se ejecuta al menos una vez.

bucle for. Bucle que repite la misma secuencia de actividades el número de veces especificado.

bucle while. Bucle que repite la misma secuencia de actividades siempre que se cumpla una condición. El bucle while comprueba su condición al principio de cada bucle. Si la condición es falsa desde el principio, la secuencia de actividades que contiene el bucle nunca se ejecuta.

bus de servicio empresarial (ESB). Infraestructura de conectividad flexible para integrar aplicaciones y servicios; ofrece un enfoque flexible y gestionable para la implementación de SOA (Service-Oriented Architecture).

bytecode. Código independiente del sistema generado por el constructor Java y ejecutado por el intérprete Java. (Sun)

calidad de servicio (QoS) . Conjunto de características de comunicación que requiere una aplicación. La calidad de servicio (QoS) define una prioridad de transmisión específica, un nivel de fiabilidad de ruta y un nivel de seguridad.

canal de contenedores web. Un tipo de canal en una cadena de transporte que crea un puente en la cadena de transporte entre un canal de entrada HTTP y un servlet o motor JSP (JSP).

canal SSL. Un tipo de canal de una cadena de transporte que asocia un repertorio de configuración SSL (Secure Sockets Layer) a la cadena de transporte.

canal TCP. Un tipo de canal de una cadena de transporte que proporciona a las aplicaciones cliente conexiones persistentes en una LAN (red de área local).

cargador. Componente que lee datos y los graba en un almacén persistente.

cargador de clases. Componente de una máquina virtual Java (JVM) responsable de buscar y cargar archivos de clase. Un cargador de clases afecta el empaquetamiento de aplicaciones y el comportamiento de aplicaciones empaquetadas desplegadas en servidores de aplicaciones.

carpeta. Un contenedor utilizado para organizar objetos.

catálogo. Un contenedor que, dependiendo del tipo de contenedor, guarda procesos, datos, recursos, organizaciones o informes en el árbol de proyecto.

categoría. Un contenedor utilizado en un diagrama de estructura para agrupar los elementos basados en un atributo o calidad compartido.

célula.

1. Grupo de procesos gestionados que están federados en el mismo gestor de despliegue y puede incluir grupos principales de alta disponibilidad.

2. Uno (o más) procesos que aloja componentes de ejecución. Cada uno tiene uno o más grupos principales a los que se ha asignado un nombre.

certificado de firmante. La entrada del certificado de confianza que generalmente está en un archivo de almacén de confianza.

certificado digital. Un documento electrónico que se utiliza para identificar a un individuo, servidor, empresa u otro tipo de entidad y para asociar una clave pública a la entidad. Los certificados digitales los emiten las autoridades certificadoras y tienen la firma digital de éstas.

chasis. El bastidor de metal en el que se montan distintos componentes electrónicos.

ciclo de vida. Pasada completa por las cuatro fases del desarrollo de software: inicio, elaboración, construcción y transición.

CIP. Véase paquete de instalación personalizado.

clase. En programación o diseño orientado a objetos, un modelo o plantilla que se puede utilizar para crear objetos con una definición común y propiedades, operaciones y comportamientos comunes. Un objeto es una instancia de una clase.

clase de bean. En Enterprise JavaBeans (EJB), clase Java que implementa una clase javax.ejb.EntityBean o una clase javax.ejb.SessionBean.

clase Java. Clase escrita en lenguaje Java.

clasificador. Un atributo especializado que se utiliza para agrupar y asignar código de color a los elementos de proceso

clave.

1. Valor matemático criptográfico que se utiliza para firmar, verificar, cifrar o descifrar digitalmente los mensajes.

2. Información que caracteriza e identifica de forma exclusiva la entidad real de cuyo seguimiento se encarga un contexto de supervisión.

clave primaria.

1. Un objeto que identifica de forma exclusiva un bean de entidad de un tipo concreto.

2. En una base de datos relacional, una clave que identifica de forma exclusiva una fila de una tabla de base de datos.

cliente. Un programa de software o un sistema que solicita servicios a un servidor. Véase también host.

cliente de aplicaciones ligero. Un tiempo de ejecución de aplicación Java, ligero y descargable, que puede interactuar con enterprise beans.

cliente ligero. Cliente que tiene instalado poco software o ninguno pero que tiene acceso a software gestionado y entregado por los servidores de red que tiene conectados. Un cliente ligero es una alternativa a un cliente de funciones completas como por ejemplo una estación de trabajo.

cliente/servidor. Relativo al modelo de interacción en el proceso de datos distribuido en el que un programa de un sistema envía una solicitud a un programa de otro sistema y espera una respuesta. El programa solicitante se denomina cliente, y el programa que responde se denomina servidor.

Cloudscape. Sistema ORDBMS (Object-Relational Database Management System), completamente Java, que puede intercalarse.

clúster. Un grupo de servidores de aplicaciones que colaboran para el equilibro de cargas de trabajo y la migración tras error.

clúster de proxy. Un grupo de servidores proxy que distribuye las solicitudes HTTP a través del clúster.

clúster de servidores. Un grupo de servidores que generalmente están en máquinas físicas diferentes y que tienen configuradas las mismas aplicaciones pero que funcionan como un servidor lógico individual.

clúster dinámico. Un clúster de servidores que utiliza pesos para equilibrar dinámicamente las cargas de trabajo de los miembros del clúster, basándose en la información de rendimiento recogida de los miembros del clúster.

código abierto. Relativo al software cuyo código fuente está disponible públicamente para utilizar o modificar. Normalmente el software de código abierto se desarrolla como colaboración pública y se pone a disposición pública, aunque su uso y redistribución pueden estar sujetos a restricciones de licencia. Linux es un ejemplo muy conocido de software de código abierto.

código de despliegue. Código adicional que permite que el código de implementación de beans escrito por un desarrollador de aplicaciones trabaje en un entorno de tiempo de ejecución EJB determinado. El código de despliegue puede generarse con herramientas que proporcione el proveedor del servidor de aplicaciones.

componente.

1. Objeto o programa reutilizable que realiza una función específica y se utiliza con otros componentes y aplicaciones.

2. En Eclipse, uno o varios plug-ins que funcionan conjuntamente para proporcionar un conjunto discreto de funciones.

componente web. Un servlet, archivo JavaServer Pages (JSP) o un archivo HTML (HyperText Markup Language). Uno o más componentes web componen un módulo web.

consulta.

1. Solicitud de información de una base de datos de acuerdo con condiciones específicas: por ejemplo, una solicitud de una lista de todos los clientes de una tabla de clientes cuyos balances son superiores a 1000 dólares.

2. Solicitud reutilizable de información sobre uno o más elementos del modelo.

consulta EJB. En lenguaje de consulta EJB, serie que contiene una cláusula SELECT opcional que especifica los objetos EJB que se deben devolver, una cláusula FROM que indica las colecciones de beans, una cláusula WHERE opcional que contiene predicados de búsqueda en las colecciones, una cláusula ORDER BY opcional que especifica la ordenación de la colección de resultados, y parámetros de entrada que corresponden a los argumentos del método buscador.

consulta SQL. Componente de determinadas sentencias SQL que especifica una tabla de resultados.

contenedor EJB. Un contenedor que implementa el contrato del componente EJB de la arquitectura Java EE. Este contrato especifica un entorno de tiempo de ejecución para los enterprise beans que incluye seguridad, simultaneidad, gestión de ciclo de vida, transacción, despliegue y otros servicios. (Sun)

contenedor web. Un contenedor que implementa el contrato del componente web de la arquitectura Java EE. (Sun)

contexto de EJB. En enterprise beans, objeto que permite que un enterprise bean invoque servicios proporcionados por el contenedor y obtenga información sobre el proceso que efectúa la llamada de un método invocado por el cliente. (Sun)

contienda de hebras. Una condición en la que una hebra está esperando un bloqueo o un objeto mantenido por otra hebra.

convertidor. En programación Enterprise JavaBeans (EJB), una clase que convierte una representación de base de datos en un tipo de objeto y viceversa.

correlación.

1. Una estructura de datos que correlaciona las claves con los valores.

2. Un archivo que define la transformación entre orígenes y destinos.

3. En el entorno de desarrollo EJB, la especificación de cómo los campos de persistencia gestionada por contenedor de un enterprise bean se corresponden a columnas en una tabla de base de datos relacional u otro almacenamiento persistente.

corriente de registro cronológico de errores. Un flujo continuado de información sobre errores que se transmite utilizando un formato definido previamente.

cortafuegos. Una configuración de red, normalmente tanto hardware como software, que impide la entrada y salida del tráfico no autorizado de una red segura.

crear una instancia. Representar una abstracción con una instancia concreta.

create, método. En enterprise beans, método definido en la interfaz inicial e invocado por el cliente para crear un enterprise bean. (Sun)

credencial. En la infraestructura JAAS (Java Authentication and Authorization Service), clase de asunto que posee atributos relacionados con la seguridad. Estos atributos pueden contener información utilizada para autenticar el sujeto en nuevos servicios.

cuadrícula de datos. Sistema que sirve para acceder a terabytes o petrabytes de datos.

cuadrícula de eXtreme Scale. Patrón que se usa para interactuar con eXtreme Scale cuando todos los datos y clientes están en un proceso.

cualificador. Un elemento simple que aporta a otro elemento simple o compuesto genérico un significado específico. Los cualificadores se utilizan en la correlación de apariciones únicas o múltiples. Un cualificador también se puede utilizar para denotar el espacio de nombres utilizado para interpretar la segunda parte del nombre, al que se hace referencia normalmente como el ID.

cuello de botella . Un punto del sistema en el que la contención de un recurso está afectando al rendimiento.

daemon. Programa que se ejecuta de manera desatendida para realizar funciones de forma continua o periódica, como el control de la red.

datos de hora de construcción. Los objetos que no son utilizados por el conversor, como los estándares EDI, los tipos de documento de datos orientados a registro, y los mapas.

DB2. Familia de programas bajo licencia de IBM para la gestión de bases de datos relacionales.

definición de documento DTD. Una descripción o diseño de un documento XML basado en un DTD XML.

definición de tipo de documento (DTD) . Reglas que especifican la estructura de una clase concreta de documentos SGML o XML. La DTD define la estructura con elementos, atributos y notaciones y establece restricciones para la utilización de cada elemento, atributo y notación en la clase particular de los documentos.

derivación. En programación orientada a objetos, el refinamiento o ampliación de una clase a partir de otra.

desalojador. Componente que controla la pertenencia de las entradas en cada instancia de BackingMap. Las memorias caché dispersas pueden utilizar desalojadores para eliminar automáticamente los datos de la memoria caché sin que ello afecte a la base de datos.

desarrollo ascendente. En los servicios web, el proceso de desarrollar un servicio a partir de un artefacto existente como por ejemplo un bean Java o un enterprise bean en vez de hacerlo a partir de un archivo WSDL (lenguaje de descripción de servicios web).

descriptor de despliegue. Archivo XML que describe cómo desplegar un módulo o aplicación especificando opciones de configuración y contenedor. Por ejemplo, un descriptor de despliegue de EJB pasa información a un contenedor EJB acerca de cómo gestionar y controlar un enterprise bean.

descubrimiento automático. El descubrimiento de artefactos de servicio en un sistema de archivos, registro externo u otra fuente.

deserialización. Un método para convertir una variable serializada en datos de objeto.

desplegable. Consulte desplegable.

desplegar. Para colocar archivos o instalar software en un entorno operativo. En Java Platform, Enterprise Edition (Java EE), esto supone crear un descriptor de despliegue adecuado al tipo de aplicación que se va a desplegar.

destino. Un punto de salida que se utiliza para entregar documentos a un sistema de programa de fondo o a un socio comercial.

destino de instalación. Sistema en el que se instalan los paquetes de instalación seleccionados.

directorio de despliegue. El directorio en el que se encuentran la configuración de servidor publicado y la aplicación web en la máquina en la que se instala el servidor de aplicaciones.

directorio LDAP. Tipo de repositorio que almacena información sobre personas, organizaciones y otros recursos, y al que se accede mediante el protocolo LDAP. Las entradas del repositorio se organizan en una estructura jerárquica, y en algunos casos dicha estructura refleja la estructura o geografía de una organización.

disparar. En la programación orientada a objetos, causar una transición de estado.

disponibilidad.

1. La condición que permite a los usuarios acceder y utilizar sus aplicaciones y datos.

2. Los periodos de tiempo durante los que se puede acceder a un recurso. Por ejemplo, una empresa puede estar disponible los días laborables de 9 AM a 5 PM y los sábados de 9 AM a 3 PM.

distribuidor de IP. Dispositivo ubicado entre las solicitudes entrantes de los usuarios y los nodos del servidor de aplicaciones que redirecciona las solicitudes a través de los nodos.

DMZ. Véase zona desmilitarizada.

DNS. Véase Sistema de nombres de dominio.

dominio. Un objeto, icono o contenedor que contiene otros objetos que representan los recursos de un dominio. Se puede utilizar el objeto de dominio para manejar estos recursos.

DTD. Véase definición de tipo de documento.

EAR. Véase archivador empresarial.

Eclipse. Iniciativa de código abierto que proporciona a los ISV y a otros desarrolladores de herramientas una plataforma estándar para elaborar herramientas de desarrollo de aplicaciones compatibles con conectores.

edición. Generación de despliegues sucesivos de un determinado conjunto de artefactos con versión.

EJB. Véase Enterprise JavaBeans.

elemento de componente. Una entidad de un componente en la que puede establecerse un punto de interrupción, como por ejemplo una actividad o un fragmento de código Java en un proceso de negocio o un primitivo de mediación o un nodo en un flujo de mediación.

elemento en espera. Una hebra que espera una conexión.

elemento global. En XML, un elemento declarado como hijo del elemento de esquema en lugar de como parte de una definición de tipo complejo. Es posible hacer referencia a los elementos globales en uno o varios modelos de contenido utilizando el atributo ref.

en desuso. Relativo a una entidad como, por ejemplo, un elemento o característica de programación a la que se da soporte pero que ya no se recomienda y puede pasar a ser obsoleta.

enlace de ámbito de célula. Ámbito de enlace en el que el enlace no es específico de ningún nodo o servidor, ni está asociado con ellos. Este tipo de enlace de nombres se crea en el contexto de raíz de persistencia de una célula.

enlace de datos JMS. Un enlace de datos que proporciona una correlación entre el formato utilizado por un mensaje JMS externo y la representación SDO (Service Data Object) utilizada por un módulo SCA (Service Component Architecture).

enlace de protocolo. Un enlace que permite al bus de servicio empresarial procesar mensajes independientemente del protocolo de comunicaciones.

en sentido ascendente. Dícese de la dirección de flujo, que va desde el inicio del proceso (arriba) hacia el final del proceso (abajo).

en sentido descendente. Dícese de la dirección de flujo, que va del primer nodo del proceso (arriba) al último nodo del proceso (abajo).

enterprise bean. Componente que implementa una tarea o una entidad de negocio y reside en un contenedor EJB. Los beans de entidad, beans de sesión y beans controlados por mensajes son enterprise beans. (Sun) Véase también bean.

Enterprise JavaBeans (EJB). Una arquitectura de componentes definida por Sun Microsystems para el desarrollo y el despliegue de aplicaciones de nivel empresarial, distribuidas y orientadas a objetos (Java EE).

entidad.

1. Clase Java sencilla que representa una fila en una tabla de base de datos o una entrada en una correlación.

2. En lenguajes de marcación como, por ejemplo, XML, colección de caracteres a la que se puede hacer referencia como una unidad, por ejemplo, para incorporar texto que se repite a menudo o caracteres especiales en un documento.

entorno. Un colección denominada de los recursos lógicos y físicos utilizados para soportar el rendimiento de una función.

entorno de despliegue. Una colección de clústeres, servidores y middleware configurados que colaboran para proporcionar un entorno para alojar módulos de software. Por ejemplo, un entorno de despliegue puede incluir un host para destinaciones de mensajes, un procesador u ordenador de sucesos de negocio y programas administrativos.

entorno de tiempo de ejecución Java. Un subconjunto de un Java developer kit que contiene los programas ejecutables básicos y los archivos que constituyen la plataforma Java estándar. JRE incluye la máquina virtual Java (JVM), las clases básicas y los archivos de soporte.

equilibrio de carga. Supervisión de los servidores de aplicaciones y gestión de la carga de trabajo en los servidores. Si un servidor sobrepasa su carga de trabajo, las solicitudes se redirigen a otro servidor con más capacidad.

error. Discrepancia entre un valor o condición calculada, observada o medida y el valor o condición cierta, especificada o teóricamente correcta.

ESB. Véase bus de servicio empresarial.

escalabilidad. La capacidad de ampliación de un sistema a medida que se van añadiendo procesadores, capacidad de memoria o capacidad de almacenamiento.

escucha. Un programa que detecta solicitudes entrantes e inicia el canal asociado.

escucha de punto final. El punto o la dirección en que un bus de integración de servicios recibe los mensajes entrantes de un servicio web.

espacio de nombres. Un contenedor lógico en el que todos los nombres son exclusivos. El identificador exclusivo para un artefacto se compone del espacio de nombres y el nombre local del artefacto.

espacio de trabajo.

1. Un directorio en el disco que contiene todos los archivos de proyecto, así como información como las preferencias.

1. Repositorio temporal de información de configuración que utilizan los clientes administrativos.

3. En Eclipse, la colección de proyectos y otros recursos que el usuario está desarrollando actualmente en el entorno de trabajo. Los metadatos relativos a estos recursos residen en un directorio del sistema de archivos; los recursos pueden residir en el mismo directorio.

esqueleto. Andamiaje de una clase de implementación.

esquema URL. Un formato que contiene la referencia de otro objeto.

estático. Una palabra clave del lenguaje de programación Java que se utiliza para definir una variable como una variable de clase.

excepción. Una condición o suceso que no puede ser manejada por un proceso normal.

exportación. Una interfaz expuesta de un módulo SCA (Service Component Architecture) que ofrece un servicio de negocio al mundo exterior. Una exportación tiene un enlace que define cómo pueden acceder al servicio los solicitantes de servicio, por ejemplo, como servicio web.

expresión. Operando SQL o XQuery o una colección de operadores y operandos SQL o XQuery que proporcionan un único valor.

eXtreme Scale distribuido. Patrón de uso para interactuar con eXtreme Scale cuando existen servidores y clientes en varios procesos.

fábrica. En programación orientada a objetos, una clase que se utiliza para crear instancias de otra clase. Las fábricas se utilizan para aislar la creación de objetos de una clase en particular en un lugar, de modo que se puedan proporcionar nuevas funciones sin cambios drásticos de código.

fábrica EJB. Bean de acceso que simplifica la creación o la búsqueda de una instancia de enterprise bean.

fase de despliegue . Véase fase de despliegue.

fase de despliegue. Una fase que incluye una combinación de crear el entorno de host para las aplicaciones y el despliegue de éstas aplicaciones. Esto incluye resolver las dependencias de recursos de las aplicaciones, las condiciones operativas, los requisitos de capacidad y las restricciones de integridad y acceso.

fixpack. Una colección acumulativa de arreglos que se pone a disposición entre paquetes de renovación planificados, renovaciones de fábrica o releases. Se tiene previsto autorizar a los clientes para desplazarse a un nivel de mantenimiento específico. Véase también arreglo temporal.

formato binario. Representación de un valor decimal en el cual todos los campos deben tener entre 2 ó 4 bytes de longitud. El signo (+ o -) se encuentra en el bit del extremo izquierdo del campo, y el valor numérico se encuentra en

los bits restantes del campo. Los números positivos tienen un 0 en el bit de signo y tienen el formato verdadero. Los números negativos tienen un 1 en el bit de signo y tienen dos formatos complementarios.

fragmento. Instancia de una partición. Un fragmento puede ser primario o una réplica.

fuga de memoria. Efecto de un programa que mantiene referencias a objetos que ya no son necesarios y que, por lo tanto, se deben reclamar.

general. Relativo a la visualización de un grupo de objetos desde un nivel alto o abstracto.

gestión de carga de trabajo. La optimización de la distribución de las solicitudes de trabajo entrantes a los servidores de aplicaciones, los enterprise beans, los servlets y otros objetos que pueden procesar la solicitud de manera más eficaz.

gestor autónomo. Conjunto de componentes de software o hardware, configurados por políticas, que gestionan el comportamiento de otros componentes de software o hardware como lo haría una persona. Un gestor autónomo incluye un bucle de control que consta de componentes de supervisión, análisis, planificación y ejecución.

Gestor de carga de trabajo (WLM). Un componente de z/OS que proporciona la capacidad de ejecutar diversas cargas de trabajo a la vez dentro de una imagen de z/OS o a lo largo de diversas imágenes.

gestor de despliegue. Un servidor que gestiona operaciones para un grupo lógico o una célula de otros servidores.

GIOP. Véase protocolo Inter-ORB general.

global.

1. Que pertenece a un elemento que está disponible para cualquier proceso del espacio de trabajo. Un elemento global aparece en el árbol de proyecto y se puede utilizar en varios procesos. Las tareas, procesos, repositorios y servicios pueden ser globales (cualquier proceso del proyecto puede hacer referencia a ellos) o locales (específicos de un solo proceso).

2. Que pertenece a la información disponible para más de un programa o subrutina.

grupo.

1. Colección de usuarios que pueden compartir autorizaciones de acceso sobre los recursos protegidos.

2. Un conjunto de documentos relacionados dentro de un intercambio. Un intercambio puede contener de cero a muchos grupos.

3. En algunos lugares, dos o más personas agrupadas por pertenecer a un lugar.

grupo HA. Una colección de uno o más miembros utilizada para proporcionar alta disponibilidad para un proceso.

HA. Véase alta disponibilidad.

hebra. Una corriente de instrucciones del sistema que controla un proceso. En algunos sistemas operativos, una hebra es la unidad de operación más pequeña en un proceso. Algunas hebras pueden ejecutarse simultáneamente llevando a cabo distintos trabajos.

herencia. Técnica de programación orientada a objetos en la que las clases existentes se usan como base para crear otras clases. Mediante herencia, los elementos más específicos incorporan la estructura y el comportamiento de los elementos más generales.

herencia EJB. Tipo de herencia en la que un enterprise bean hereda propiedades, métodos y atributos del descriptor de control a nivel de método de otro enterprise bean que reside en el mismo grupo.

High Availability Manager. Una infraestructura en la que se determina la pertenencia de los miembros al grupo principal y se comunica el estado entre los miembros del grupo principal.

host.

1. Sistema que está conectado a una red y proporciona un punto de acceso a esa red. El host puede ser un cliente, un servidor, o ambos simultáneamente.

2. En los perfiles de rendimiento, una máquina que es propietaria de procesos de los que se están creando perfiles. Véase también servidor.

host virtual. Una configuración que permite que un host parezca varios hosts. Los recursos asociados con un host virtual no pueden compartir datos con recursos asociados con otro host virtual, incluso si los hosts virtuales comparten la misma máquina física.

HTTPS.

1. Véase HTTP sobre SSL.

2. Véase protocolo seguro de transferencia de hipertexto.

HTTP sobre SSL (HTTPS) . Un protocolo web para transacciones seguras que cifra y descifra solicitudes de página de usuario y páginas devueltas por el servidor web.

IDE. Siglas de Integrated Development Environment. Véase Entorno de Desarrollo Integrado.

identificador de instancia global. Identificador exclusivo globalmente que lo genera la aplicación o el emisor y que se utiliza como clave primaria para la identificación del suceso.

Identificador universal de recursos (URI) .

1. Una serie compacta de caracteres para identificar un recurso abstracto o físico.

2. Una dirección exclusiva que se utiliza para identificar el contenido en la web como, por ejemplo, una página de texto, un vídeo o un clip de sonido, una imagen estática o animada o un programa. El formato más común del URI es la dirección de página web, que es una forma o subconjunto concreto de URI denominado localizador universal de recursos (URL). Un URI describe habitualmente cómo acceder al recurso, el sistema que contiene el recurso y el nombre del recurso (un nombre de archivo) en el sistema.

IIOP. Véase protocolo Inter-ORB de Internet.

importación.

1. Un artefacto de desarrollo que importa un servicio que es externo a un módulo.

2. El punto en el que un módulo SCA accede a un servicio externo (un servicio fuera del módulo SCA) como si fuera local. Una importación define las interacciones entre el módulo SCA y el proveedor de servicios. Una importación tiene un enlace y una o más interfaces.

índice. Conjunto de punteros que están ordenados lógicamente según los valores de una clave. Los índices proporcionan acceso rápido a los datos e imponen la exclusividad de los valores clave de las filas de la tabla.

Information Center. Recopilación de información que proporciona soporte a los usuarios de uno o más productos, puede iniciarse independientemente desde el producto e incluye una lista de temas para la navegación y un motor de búsqueda.

informe autorizado de análisis de programa (APAR) . Una solicitud de corrección de un defecto en un release soportado de un programa proporcionado por IBM.

instalación silenciosa. Instalación que no envía mensajes a la consola sino que almacena los mensajes y errores en archivos de registro. Una instalación silenciosa puede utilizar archivos de respuesta para la entrada de datos.

instancia. Una aparición específica de un objeto que pertenece a una clase.

instancia de componente. Un componente de ejecución que puede ejecutarse en paralelo con otras instancias del mismo componente.

Integrated Development Environment (IDE) . Conjunto de herramientas de desarrollo de software, como los editores del fuente, los compiladores y los depuradores, a las que se puede acceder desde una sola interfaz de usuario.

interfaz. Una colección de operaciones que se utilizan para especificar un servicio de una clase o componente.

interfaz de programación de aplicaciones (API) . Una interfaz que permite que un programa de aplicación escrito en un lenguaje de nivel alto pueda utilizar funciones o datos específicos del sistema operativo o de otro programa.

Interfaz Profiler de la máquina virtual Java (JVMPI). Una herramienta de perfiles que da soporte a la recopilación de información como, por ejemplo, datos sobre la recopilación de errores, y la API de JVM (máquina virtual Java) que ejecuta el servidor de aplicaciones.

Intermediario de solicitud de objetos (ORB) . En programación orientada a objetos, el software que sirve de intermediario al habilitar objetos de forma transparente para el intercambio de solicitudes y respuestas.

invocación. La activación de un programa o procedimiento.

IP. Siglas de Internet Protocol. Véase protocolo Internet.

JAAS. Véase Java Authentication and Authorization Service.

JAF. Véase JavaBeans Activation Framework.

Java. Lenguaje de programación orientado a objetos para código de interpretación portable que soporta la interacción entre objetos remotos. Java fue desarrollado y especificado por Sun Microsystems, Incorporated.

Java Authentication and Authorization Service (JAAS). En la tecnología Java EE, una API estándar para realizar operaciones basadas en seguridad. A través de JAAS, los servicios pueden autenticar y autorizar usuarios al tiempo que permiten que las aplicaciones sigan siendo independientes de las tecnologías subyacentes.

JavaBeans. Según la definición de Sun Microsystems para Java, modelo de componente portable, independiente de la plataforma y reutilizable. Véase también bean.

JavaBeans Activation Framework (JAF). Una ampliación estándar de la plataforma Java que determina los tipos de datos arbitrarios y las operaciones disponibles y que puede crear una instancia de bean para ejecutar los servicios pertinentes.

Java Database Connectivity (JDBC). Estándar del sector para la conectividad independiente de la base de datos entre la plataforma Java y una amplia gama de bases de datos. La interfaz JDBC proporciona una interfaz a nivel de llamada para el acceso a bases de datos basadas en SQL y basadas en XQuery.

Javadoc.

1. Una herramienta que analiza las declaraciones y los comentarios de documentación en un conjunto de archivos de origen y genera un conjunto de páginas HTML que describen clases, clases internas, interfaces, constructores, métodos y campos. (Sun)

2. Pertenece a la herramienta que analiza las declaraciones y los comentarios de documentación en un conjunto de archivos de origen y genera un conjunto de páginas HTML que describen las clases, clases internas, interfaces, constructores, métodos y campos.

Java EE. Véase Java Platform, Enterprise Edition.

Java EE Connector Architecture (JCA). Arquitectura estándar para conectar la plataforma Java EE a sistemas de información de empresa heterogéneos (EIS).

Java Management Extensions (JMX). Un medio de realizar la gestión mediante tecnología Java. JMX es una ampliación abierta y universal del lenguaje de programación Java para la gestión y se puede desplegar en todos los sectores en los que ésta sea necesaria.

Java Message Service (JMS). Interfaz de programas de aplicación que proporciona funciones de lenguaje Java para manejar mensajes.

Java Naming and Directory Interface (JNDI). Ampliación de la plataforma Java que proporciona una interfaz estándar para los servicios de denominación y directorio heterogéneos.

Java Platform, Enterprise Edition (Java EE). Un entorno para desarrollar y desplegar aplicaciones empresariales, definido por Sun Microsystems Inc. La plataforma Java EE consta de un conjunto de servicios, interfaces de programación de aplicaciones (API) y protocolos que proporcionan la funcionalidad para desarrollar aplicaciones basadas en la web de varios niveles. (Sun)

Java Platform, Standard Edition (Java SE). Plataforma central de la tecnología Java. (Sun)

JavaScript. Un lenguaje de creación de scripts web que se utiliza en navegadores y servidores web. (Sun)

JavaScript Object Notation. Un formato de intercambio de datos ligero que se basa en la notación literal de objetos de JavaScript. JSON es independiente del lenguaje de programación, sin embargo utiliza convenciones de lenguajes entre los que se incluyen C, C++, C#, Java, JavaScript, Perl, Python.

Java SE. Véase Java Platform, Standard Edition.

Java Secure Socket Extension (JSSE). Un paquete Java que permite las comunicaciones seguras a través de Internet. Implementa una versión Java de los protocolos SSL (Secure Sockets Layer) y TLS (Transport Layer Security) y da soporte al cifrado de datos, la autenticación del servidor, la integridad de mensajes y, opcionalmente, la autenticación de clientes.

Java SE Development Kit (JDK). El nombre del kit de desarrollo de software que Sun Microsystems proporciona para la plataforma Java.

JavaServer Pages (JSP). Tecnología de scripts del lado del servidor que permite que el código Java se intercale dinámicamente en las páginas web (archivos HTML) y se ejecute en el momento de servir la página, con objeto de devolver contenido dinámico a un cliente.

Java Specification Request (JSR). Una especificación propuesta formalmente para la plataforma Java.

JAX. Siglas de Java API for XML. Véase API de Java para XML.

JCA . Véase Java EE Connector Architecture.

JDBC. Véase Java Database Connectivity.

JDK. Véase Java SE Development Kit.

jerarquía de clases. Las relaciones entre las clases que comparten una única herencia.

JMS. Véase Java Message Service.

JMX. Véase Java Management Extensions.

JNDI. Véase Java Naming and Directory Interface.

JSP. Véase JavaServer Pages.

JSR. Siglas de Java Specification Request. Véase petición de especificación Java (JSR).

JSSE. Véase Java Secure Socket Extension.

JVM. Siglas de Java Virtual Machine. Véase máquina virtual Java (JVM).

JVMPI. Véase Java virtual machine Profiler Interface.

Jython. Una implementación del lenguaje de programación Python integrado en la plataforma Java.

kit de desarrollo de software (SDK) . Un conjunto de herramientas, API, documentación para ayudar en el desarrollo de software de un lenguaje de sistemas específico o de un entorno operativo determinado.

LDAP. Véase protocolo LDAP (Lightweight Directory Access Protocol).

lectura no permitida. Una solicitud de lectura que no implica ningún mecanismo de bloqueo. Esto significa que los datos que se pueden leer se podrían retrotraer más adelante y generar una incoherencia entre lo que se ha leído y lo que está en la base de datos.

lenguaje de códigos ampliable (XML) . Metalenguaje estándar para definir lenguajes de marcación basado en SGML (Standard Generalized Markup Language).

Lenguaje de consulta estructurado (SQL) . Un lenguaje estándar para definir y manipular los datos en una base de datos relacional.

lenguaje de mandatos Java. Un lenguaje de scripts para el entorno Java que se utiliza para crear contenidos web y controlar las aplicaciones Java.

Lightweight Directory Access Protocol (LDAP). Un protocolo abierto que utiliza TCP/IP para proporcionar acceso a directorios que admiten el modelo X.500 y que no acarrea los requisitos de recursos del protocolo más complejo DAP (Directory Access Protocol) de X.500. Por ejemplo, LDAP se puede utilizar para localizar personas, organizaciones y otros recursos de directorios de Internet o intranet.

Lightweight Third Party Authentication (LTPA). Un protocolo que utiliza la criptografía para dar soporte a la seguridad en un entorno distribuido.

línea de mandatos. La línea en blanco en una visualización donde se pueden introducir mandatos, números de opción o selecciones.

local.

1. Relativo a un dispositivo, archivo o sistema al que se accede directamente desde un sistema de usuario, sin el uso de una línea de comunicaciones.

2. Relativo a un elemento que sólo está disponible en su propio proceso.

Localizador universal de recursos (URL) . Dirección exclusiva de un recurso de información que sea accesible en una red como Internet. El URL incluye el nombre abreviado del protocolo utilizado para acceder al recurso de información y la información utilizada por el protocolo para localizar el recurso de información.

Lo que se ve es lo que se obtiene (WYSIWYG) . Capacidad de un editor para visualizar de forma continuada las páginas exactamente igual a como se imprimirán o representarán.

LTPA. Véase Lightweight Third Party Authentication.

manejador de excepciones. Un conjunto de rutinas que responde a una condición anómala. Un manejador de excepciones puede interrumpir y reanudar la normal ejecución de los procesos.

máquina virtual. Una especificación abstracta de un dispositivo informático que se puede implementar de varias formas en el software y el hardware.

Máquina virtual Java (JVM). Implementación de software de un procesador que ejecuta código Java compilado (applets y aplicaciones).

MBean. Véase bean gestionado.

memoria caché coherente. Memoria caché que conserva la integridad de modo que todos los clientes ven los mismos datos.

memoria caché de grabación a través. Memoria caché que graba de forma síncrona cada operación de grabación en la base de datos mediante un cargador.

memoria caché de grabación diferida. Memoria caché que graba de forma asíncrona cada operación de grabación en la base de datos mediante un cargador.

memoria caché de lectura a través. Memoria caché dispersa que carga entradas de datos por clave según se soliciten. Si no se encuentran los datos en la memoria caché, los datos que faltan se recuperan mediante el cargador, que carga los datos del repositorio de datos de fondo y los inserta en la memoria caché.

memoria caché dinámica. Una consolidación de varias actividades de colocación en memoria caché, incluidos los servlets, los servicios web y los mandatos de WebSphere, en un servicio donde estas actividades comparten parámetros de configuración y trabajan conjuntamente para mejorar el rendimiento.

mensajería asíncrona. Método de comunicación entre programas en el que un programa coloca un mensaje en una cola de mensajes y, a continuación, continúa con su propio proceso sin esperar una respuesta a su mensaje.

mensajería gestionada por bean. Una función de mensajería asíncrona que proporciona a un enterprise bean control total sobre la infraestructura de mensajes.

método. En la programación orientada a objetos, una operación que un objeto puede realizar. Un objeto puede tener muchos métodos.

método de establecimiento. Un método cuya finalidad es establecer el valor de una instancia o variable de clase. Esta posibilidad permite a otro objeto establecer el valor de una de sus variables.

método de obtención. Método cuya finalidad es obtener el valor de una instancia o variable de clase. Esto permite a otro objeto averiguar el valor de una de sus variables.

métrica. Un poseedor de información, normalmente una magnitud de rendimiento empresarial, en un contexto de supervisión.

migración tras error. Una operación automática que conmuta a un sistema redundante o en espera en caso de que se produzca una interrupción en el software, el hardware o la red.

modalidad de mantenimiento. Estado de un nodo o servidor que un administrador puede utilizar para diagnosticar, mantener o ajustar el nodo o servidor sin interrumpir el tráfico entrante en un entorno de producción.

modalidad silenciosa. Un método para instalar o desinstalar un componente de producto desde la línea de mandatos sin interfaz gráfica. Cuando se utiliza la modalidad silenciosa, se especifican los datos necesarios para el programa de instalación o desinstalación directamente en la línea de mandatos o en un archivo (denominado archivo de opciones o archivo de respuestas).

módulo EJB. Unidad de software que consta de uno o más enterprise beans y un descriptor de despliegue de EJB (Sun)

navegador web. Un programa cliente que inicia solicitudes en un servidor web y visualiza la información que devuelve el servidor.

nodo.

1. Una agrupación lógica de servidores gestionados.

2. Cualquier elemento de un control de árbol, incluidos un elemento sencillo, elemento compuesto, mandato de correlación, comentario o nodo de grupos.

3. En XML, la unidad más pequeña de una estructura válida y completa en un documento.

4. Las formas fundamentales que conforman un diagrama.

nodo hijo. Un nodo que se encuentra dentro del ámbito de otro nodo.

nombre de host.

´1. En la comunicación por Internet, nombre asignado a un sistema. El nombre de host podría ser un nombre de dominio totalmente calificado como, por ejemplo, mycomputer.city.company.com, o podría ser un subnombre específico como mycomputer.

2. Nombre de red de un adaptador de red en una máquina física donde está instalado el nodo.

nombre largo. La propiedad que especifica el nombre lógico del servidor en la plataforma z/OS.

Nombre uniforme de recursos (URN) . Nombre que identifica de forma exclusiva un servicio web ante un cliente.

número de puerto. En las comunicaciones por Internet, el identificador de un conector lógico entre una entidad de aplicación y el servicio de transporte.

ObjectGrid. Base de datos de memoria compatible con cuadrículas para las aplicaciones escritas en Java. ObjectGrid se puede utilizar como una base de datos en memoria o para distribuir datos en una red.

objeto. En un diseño o una programación orientados al objeto, una realización concreta (instancia) de una clase que está formada por datos y las operaciones asociadas a dichos datos. Un objeto contiene los datos de la instancia que define la clase pero ésta es la propietaria de las operaciones asociadas a los datos.

objeto EJB. En enterprise beans, un objeto cuya clase implementa la interfaz remota del enterprise bean (Sun).

objeto genérico. Un objeto que se utiliza en llamadas de API y expresiones XPATH para hacer referencia a conceptos, entidades personalizadas o colecciones. Por ejemplo, la expresión XPATH /WSRR/GenericObject recupera todos los conceptos de WebSphere Service Registry and Repository.

objeto inicial EJB. En programación de Enterprise JavaBeans (EJB), un objeto que proporciona las operaciones de ciclo de vida (crear, eliminar, buscar) para un enterprise bean. (Sun)

ODBC. Véase Open Database Connectivity.

Open Database Connectivity (ODBC). Una interfaz de programación de aplicaciones (API) estándar para el acceso de datos en sistemas de gestión de bases de datos tanto relacionales como no relacionales. A través del uso de esta API, las aplicaciones de base de datos pueden acceder a datos almacenados en sistemas de gestión de bases de datos en una serie de sistemas, incluso, aunque el sistema de gestión de bases de datos utilice un formato diferente de almacenamiento de datos y una diferente interfaz de programación.

operación. Una implementación de funciones o consultas que un objeto puede realizar.

ORB. Siglas de Object Request Broker. Véase intermediario de solicitud de objetos.

organización. Una entidad donde las personas colaboran para conseguir objetivos específicos, como por ejemplo una empresa, una compañía o una fábrica.

página JSP. Un documento basado en texto que utiliza datos de plantillas fijos y elementos JSP que describe cómo procesar una solicitud para crear una respuesta. (Sun)

palabra clave. Una de las palabras predefinidas de un lenguaje de programación, un lenguaje de artificial, una aplicación o un mandato.

panel de instrumentos. Una página web que puede contener uno o más visores que representan gráficamente datos de negocio.

paquete.

1. En programación Java, un grupo de tipos. Los paquetes se declaran con la palabra clave de paquete. (Sun)

2. Envoltura alrededor del contenido del documento que define el formato utilizado para transmitir un documento por Internet, por ejemplo, RNIF, AS1 y AS2.

3. Ensamblar componentes en módulos y módulos en aplicaciones empresariales.

paquete de instalación. Una unidad instalable de un producto de software. Los paquetes de producto de software son unidades instalables de forma separada que pueden funcionar de forma independiente de otros paquetes de dicho producto de software.

paquete de instalación personalizado (CIP). Imagen de instalación personalizada que puede incluir uno o más paquetes de mantenimiento, un archivo archivador de configuración de un perfil de servidor autónomo, uno o más archivos archivadores de empresa, scripts y otros archivos que ayudan a personalizar la instalación resultante.

paquete de renovación. Un conjunto acumulativo de arreglos que contiene funciones nuevas. Véase también fixpack, arreglo interino.

perfil. Los datos que describen las características de un usuario, grupo, programa, dispositivo o ubicación remota.

Performance Monitoring Infrastructure (PMI). Un conjunto de bibliotecas y paquetes asignados para recopilar, entregar, procesar y mostrar datos de rendimiento.

permiso. Autorización para realizar actividades como, por ejemplo, leer y escribir en archivos locales, crear conexiones de red y cargar el código nativo.

persistencia.

1. Una característica de datos que se mantienen entre los límites de sesión, o de un objeto que sigue existiendo después de la ejecución del programa o proceso que lo ha creado, normalmente, en almacenamiento volátil, como un sistema de base de datos.

2. En Java EE, protocolo para transferir el estado de un bean de entidad entre sus variables de instancia y una base de datos subyacente. (Sun)

persistencia gestionada por bean (BMP) . El mecanismo en el que el bean de entidad gestiona la transferencia de datos entre las variables de un bean de entidad y un gestor de recursos. (Sun)

persistir. Para mantenerse más allá de los límites de la sesión, generalmente, en almacenamiento no volátil, por ejemplo, en un sistema de base de datos o un directorio.

pila. Área de la memoria en la que se almacena información como la información temporal de registro, los valores de los parámetros y las direcciones de retorno de las subrutinas y que se basa en el principio de que el último en entrar es el primero en salir (LIFO - last in, first out).

plan de construcción. Un archivo XML que define el proceso necesario para construir salidas de generación y que especifica el sistema en el que tiene lugar el proceso.

plataforma Java. Un término colectivo del lenguaje Java para escribir programas; un conjunto de API, bibliotecas de clases y otros programas utilizados para desarrollar, compilar y comprobar errores en programas y una máquina virtual Java que carga y ejecuta los archivos de clases. (Sun)

plug-in. Un módulo de software que se puede instalar por separado que añade funcionalidad a un programa, aplicación o interfaz existente.

plug-in de servidor web. Un módulo de software que da soporte al servidor web para la comunicación de solicitudes de contenido dinámico, por ejemplo, servlets, con el servidor de aplicaciones.

PMI. Véase Performance Monitoring Infrastructure.

política. Conjunto de consideraciones que influyen en el comportamiento de un recurso gestionado o de un usuario.

política de autorización. Una política cuyo destino es un servicio de negocio y cuyo contrato contiene una o más aserciones que otorgan permiso para ejecutar una acción de canal.

política de despliegue. Modo opcional de configurar un entorno eXtreme Scale basado en varios elementos: número de sistemas, servidores, particiones, réplicas (incluido el tipo de réplica) y tamaños de almacenamiento dinámico de cada servidor.

política HA. Un conjunto de reglas que se define para un grupo HA que dicta si se activan cero (0) o más miembros. La política se asocia a un grupo HA específico haciendo coincidir el criterio de coincidencia de políticas con el nombre de grupo.

preciso. Relativo a la visualización en detalle de un objeto individual.

proceso.

1. Procedimiento progresivo y continuado que consta de una serie de actividades controladas que se dirigen sistemáticamente hacia un resultado o fin concreto.

2. Secuencia de documentos o mensajes que se van a intercambiar entre los gestores de comunidad y los participantes para ejecutar una transacción de negocio.

proceso síncrono. Proceso que se inicia invocando una operación de respuestas a solicitudes. Resultado del proceso devuelto por la misma operación.

programación orientada a objetos. Un sistema de programación basado en el concepto de abstracción de datos y herencia. Al contrario que las técnicas de programación de procedimiento, la programación orientada a objetos no se concentra en cómo se consigue algo sino en qué objetos abarcan el problema y cómo se manipulan.

programa de arranque. Un pequeño programa que carga programas más grandes durante la inicialización.

propiedad. Una característica de un objeto que describe el objeto. Una propiedad se puede cambiar o modificar. Las propiedades pueden describir un nombre de objeto, tipo, valor o comportamiento, entre otras cosas.

Protocolo de control de transmisiones/protocolo Internet (TCP/IP) . Un conjunto de protocolos de comunicaciones estándar de la industria y no propietario que proporciona conexiones fiables de extremo a extremo entre aplicaciones a través de redes interconectadas de distintos tipos.

Protocolo de control de transmisiones (TCP) . Protocolo de comunicaciones empleado en Internet y en cualquier red que esté en conformidad con los estándares de Internet Engineering Task Force (IETF) como protocolo entre redes. TCP proporciona un protocolo fiable de host a host de las redes de comunicaciones de paquetes conmutados y de los sistemas interconectados de dichas redes.

Protocolo Internet (IP) . Un protocolo que direcciona datos a través de una red o de redes interconectadas. Este protocolo actúa como intermediario entre las capas de protocolo superiores y la red física.

Protocolo Inter-ORB de Internet (IIOP) . Protocolo que se utiliza para establecer comunicación entre los intermediarios de solicitud de objetos CORBA (Common Object Request Broker Architecture).

protocolo Inter-ORB general (GIOP) . Un protocolo que utiliza CORBA (Common Object Request Broker Architecture) para definir el formato de los mensajes.

Protocolo seguro de transferencia de hipertexto (HTTPS) . Un protocolo de Internet que utilizan los servidores web y los navegadores web para transferir y visualizar documentos de hipermedia (hipertexto y multimedia) de modo seguro a través de Internet.

proveedor de MBean. Biblioteca que contiene una implementación de un MBean JMX (Java Management Extensions) y su archivo descriptor XML (Extensible Markup Language) de MBean.

proxy. Una pasarela de aplicación de una red a otra para una aplicación de red específica como, por ejemplo, Telnet o FTP, por ejemplo, donde un servidor de Telnet de proxy de cortafuegos realiza una autenticación del usuario y, a continuación, permite que el tráfico fluya a través del proxy, como si no estuviera. La función se realiza en el cortafuegos y no en la estación de trabajo del cliente, lo que supone más carga de trabajo en el cortafuegos.

proyecto de aplicación empresarial (proyecto EAR) . Estructura y jerarquía de carpetas y archivos que contienen un descriptor de despliegue y un documento de ampliación de IBM, así como los archivos que son comunes a todos los módulos Java EE definidos en el descriptor de despliegue.

proyecto EAR. Véase proyecto de aplicación empresarial.

proyecto EJB. Un proyecto que contiene los recursos que se necesitan para las aplicaciones EJB, incluidos los enterprise beans, las interfaces inicial, local y remota, los archivos JSP, los servlets y los descriptores de despliegue.

proyecto Java. En Eclipse, proyecto que contiene código fuente Java compilable y que funciona a modo de contenedor de paquetes o carpetas fuente.

prueba de componentes. Una prueba automatizada de uno o varios componentes de una aplicación de negocio que puede incluir clases Java, beans EJB o servicios web.

PTF. Siglas de Program Temporary Fix. Véase arreglo temporal de programa.

público.

1. En la programación orientada a objetos, relativo a un miembro de clase que es accesible a todas las clases.

2. En el lenguaje de programación Java, relativo a un método o variable a la que pueden acceder elementos que residen en otras clases. (Sun)

puerto. Como se ha definido en un documento WSDL (lenguaje de descripción de servicios web), un único punto final definido como una combinación de un enlace y una dirección de red.

puerto de escucha. Objeto que define la asociación entre una fábrica de conexiones, un destino y un bean desplegado controlado por mensaje. Los puertos de escucha simplifican la administración de las asociaciones entre estos recursos.

pulsación. Una señal que una entidad envía a otra para indicar que sigue activa.

punto a punto. Relativo a un tipo de aplicación de mensajería en la que la aplicación emisora conoce la destinación del mensaje.

punto de acceso de igual proxy. Un medio de identificar los valores de comunicaciones para un punto de acceso de igual al que no se puede acceder directamente.

punto de interrupción. Un punto marcado en un proceso o flujo programático que provoca que el flujo se interrumpa cuando se alcanza el punto, normalmente para poder depurar o supervisar.

punto de interrupción de entrada. Un punto de interrupción establecido sobre un elemento de componente que se alcanza antes de invocar el elemento de componente.

punto final.

1. Una aplicación CA u otro consumidor de cliente de un suceso del sistema de información empresarial.

2. El sistema que es el origen o destino de una sesión.

punto muerto. Una condición en la que se bloquean dos hebras independientes de control, cada una espera a que la otra emprenda alguna acción. Un punto muerto suele aparecer al añadir mecanismos de sincronización para evitar condiciones de actualización.

QoS. Siglas de Quality of Service. Véase calidad de servicio.

rastreador web. Un tipo de rastreador que explora la web recuperando un documento web y siguiendo los enlaces dentro de ese documento.

rastreo de ejecución. Una cadena de sucesos que se registra y se visualiza en formato jerárquico en la página Sucesos del cliente de prueba de integración.

recogida de basura. Una rutina de búsqueda en la memoria para reclamar espacio de segmentos de programas o datos inactivos.

recurrencia. Técnica de programación en la que un programa (o rutina) se llama a sí mismo para realizar los pasos sucesivos de una operación, en la que cada paso utiliza la salida del paso precedente.

recurso.

1. Un activo discreto, por ejemplo, conjuntos de aplicaciones, aplicaciones, servicios empresariales, interfaces, puntos finales y sucesos de negocio.

2. Un recurso de un sistema o sistema operativo necesario para un trabajo, una tarea o un programa en ejecución. Los recursos incluyen el almacenamiento principal, dispositivos de entrada/salida, la unidad de proceso, conjuntos de datos, archivos, bibliotecas, carpetas, servidores de aplicaciones, y programas de control o de proceso.

3. Una persona, parte de un equipo o material que se utiliza para realizar una tarea o proyecto. Cada recurso es una instancia o ejemplo concreto de una definición de recurso.

recurso de instancia de memoria caché. Una ubicación en la que cualquier aplicación Java Platform, Enterprise Edition (Java EE) puede almacenar, distribuir y compartir datos.

recurso de particionamiento. Infraestructura de programación e infraestructura de gestión de sistemas que soporta el concepto de particionamiento para enterprise beans, el tráfico HTTP y el acceso a bases de datos.

referencia de EJB. Un nombre lógico utilizado por una aplicación para ubicar la interfaz inicial de un enterprise bean en el entorno operativo destino.

región. Un área contigua de almacenamiento virtual que tiene características comunes que se pueden compartir entre procesos.

región de servicio. Un área contigua del almacenamiento virtual que se inicia dinámicamente cuando aumenta la carga y que se detiene automáticamente cuando decrece la carga.

registro cronológico. El registro de datos acerca de sucesos específicos del sistema como, por ejemplo, errores.

regla if-then. Regla en la que la acción (parte then) sólo se realiza cuando se cumple la condición (parte if).

rendimiento . La medida de la cantidad de trabajo realizado por un dispositivo como un sistema o una impresora, durante un período de tiempo; por ejemplo, el número de trabajos por día.

repetición. Véase bucle.

repetidor. Una clase o construcción que se utiliza para revisar una colección de objetos a la vez.

réplica. El proceso de mantener un conjunto definido de datos en más de una ubicación. La réplica supone la copia de cambios designados de una ubicación (origen) a otra (destino) y la sincronización de los datos en ambas ubicaciones.

réplica. Servidor que contiene una copia del directorio o directorios de otro servidor. Las réplicas realizan una copia de seguridad para mejorar el rendimiento y los tiempos de respuesta y para garantizar la integridad de los datos.

réplica asíncrona. Fragmento que recibe actualizaciones después de la confirmación de la transacción. Este método es más rápido que la réplica síncrona, pero puede haber pérdida de datos porque la réplica asíncrona puede estar varias transacciones por detrás del fragmento primario.

réplica de memoria caché. La compartición de los ID, las entradas y las invalidaciones de memoria caché con otros servidores del mismo dominio de réplica.

réplica síncrona. Fragmento que recibe actualizaciones como parte de la transacción en el fragmento primario para garantizar la coherencia de los datos, lo cual puede aumentar el tiempo de respuesta en comparación con la réplica asíncrona.

restricción temporal. Una acción de validación utilizada para medir la duración de una llamada de método o de una secuencia de llamadas de método.

rol.

1. Una descripción de una función que debe llevar a cabo un recurso individual o masivo, y las calificaciones necesarias para cumplir la función. En simulación y análisis, el término rol se utiliza también para referirse a los recurso cualificados.

2. Una función de trabajo que identifica las tareas que puede realizar un usuario y los recursos a los que tiene acceso un usuario. A un usuario se le pueden asignar uno o más roles.

3. Un grupo lógico de principales que proporciona un conjunto de permisos. El acceso a las operaciones se controla concediendo acceso a un rol.

4. En una relación un rol determina la función y la participación de las entidades. Los roles recogen los requisitos de estructura y restricciones de las entidades participantes y su forma de participación. Por ejemplo, en una relación de contratación, los roles son empresario y empleado.

root. El nombre de usuario del usuario del sistema con más autoridad.

rutina de carga. El proceso mediante el cual se obtiene una referencia inicial del servicio de denominación. El valor de rutina de carga y el nombre de host constituyen el contexto inicial de las referencias JNDI (Java Naming and Directory Interface).

salud. Condición o estado general del entorno de la base de datos.

script. Un estilo de programación que reutiliza los componentes existentes como base para crear aplicaciones.

script. Serie de mandatos, combinados en un archivo, que llevan a cabo una función concreta cuando se ejecuta el archivo. Los scripts se interpretan a medida que se ejecutan.

script de shell. Un programa o script que es interpretado por el shell de un sistema operativo.

SDK. Siglas de Software Development Kit. Véase kit de desarrollo de software (SDK).

Secure Sockets Layer (SSL). Protocolo de seguridad que proporciona privacidad de comunicación. Con SSL, las aplicaciones de cliente/servidor pueden comunicarse de una forma diseñada para impedir las escuchas no deseadas, la manipulación indebida y la falsificación.

seguridad global . Relativa a todas las aplicaciones que se ejecutan en el entorno, y determina si se utiliza algún tipo de seguridad, el tipo de registro que se utiliza para la autenticación, y otros valores, la mayoría de los cuales son valores por omisión.

seguridad Java Connector. Una arquitectura diseñada para ampliar el modelo de seguridad de extremo a extremo para que las aplicaciones basadas en Java EE incluyan sistemas de información empresarial (EIS).

señal.

1. Un marcador utilizado para rastrear el estado actual de la instancia del proceso durante una ejecución de simulación.

2. Un mensaje determinado o patrón de bits que indica un permiso o control temporal de transmisión en una red.

señal de seguridad. Una representación de un conjunto de reclamaciones que puede realizar un cliente y que puede incluir un nombre, contraseña, identidad, clave, certificado, grupo, privilegio, etc.

separación del servidor web. Una topología en la que el servidor web está separado físicamente del servidor de aplicaciones.

serialización. En programación orientada a objetos, la grabación de datos de modo secuencial en un soporte de comunicaciones desde la memoria de programa.

serializador. Un método para convertir datos de objeto a otro formato, como por ejemplo binario o XML.

serie. En los lenguajes de programación, la forma de datos utilizada para almacenar y manipular texto.

servicio de catálogo. Servicio que controla la ubicación de fragmentos y descubre y supervisa la salud de los contenedores.

servidor. Un programa de software o un sistema que proporciona servicios a otros programa de software u otros sistemas. Véase también host.

servidor autónomo. Un servicio de catálogo o servidor de contenedor que se gestiona desde el sistema operativo que inicia y detiene el proceso del servidor.

servidor de aplicaciones. Un programa servidor en una red distribuida que proporciona el entorno de ejecución para un programa de aplicación.

servidor de contenedor. Instancia de servidor que puede albergar varios fragmentos. Una máquina virtual Java (JVM) puede albergar varios servidores de contenedor.

servidor de supervisión de TCP/IP. Entorno de tiempo de ejecución que supervisa todas las solicitudes y respuestas entre el navegador web y un servidor de aplicaciones así como la actividad de TCP/IP.

servidor EJB. Software que proporciona servicios a contenedores de EJB. Un servidor EJB puede albergar uno o más contenedores EJB. (Sun)

servidor incorporado. Servicio de catálogo o servidor de contenedor que reside en un proceso existente y se inicia y detiene dentro del proceso.

servidor Java EE. Un entorno de tiempo de ejecución que proporciona contenedores EJB o web.

servidor proxy.

1. Un servidor que actúa como un intermediario para las solicitudes web HTTP que se incluyen en una aplicación o un servidor web. Un servidor proxy actúa como sustituto de los servidores de contenido de la empresa.

2. Un servidor que recibe peticiones previstas para otro servidor y que actúa en nombre del cliente (como el proxy del cliente) para obtener el servicio solicitado. Un servidor proxy se utiliza a menudo cuando el cliente y el servidor no son compatibles para la conexión directa. Por ejemplo, el cliente no puede cumplir los requisitos de autenticación de seguridad del servidor pero debe poder acceder a algunos servicios.

servidor web. Programa de software que puede dar servicio a las solicitudes HTTP (Hypertext Transfer Protocol).

servlet. Un programa Java que se ejecuta en un servidor web y amplía las funciones del servidor generando contenido dinámico en respuesta a las peticiones del cliente web. Los servlets se utilizan generalmente para conectar las bases de datos a la web.

sesión.

1. Conexión lógica o virtual entre dos estaciones, programas de software o dispositivos de una red que permite que los dos elementos se comuniquen e intercambien datos.

2. Una serie de solicitudes a un servlet que se origina a partir del mismo usuario en el mismo navegador.

3. En Java EE, un objeto utilizado por un servlet para rastrear la interacción del usuario con una aplicación web entre varias peticiones HTTP.

sincronizar. Añadir, restar o cambiar una característica o un artefacto para hacerlo coincidir con otro.

sintaxis. Reglas para la creación de un mandato o una sentencia.

Sistema de nombres de dominio (DNS). Sistema de base de datos distribuida que correlaciona nombres de dominio con direcciones IP.

sistema host. Un sistema informático que es un sistema principal de empresa que alberga las aplicaciones 3270. En las herramientas de desarrollo de servicio terminal 3270, el desarrollador utiliza el grabador de servicio terminal 3270 para conectar con el host.

sitio web. Un conjunto relacionado de archivos disponibles en la web gestionado por una entidad única (una organización o un individuo) y contiene información en hipertexto para sus usuarios. Un sitio web incluye a menudo enlaces de hipertexto a otros sitios web.

SLA. Siglas de Service Level Agreement. Véase acuerdo de nivel de servicio.

solicitud. Un componente de una acción que indica que la entrada de usuario es necesaria para un campo antes de pasar a una pantalla de salida.

soporte basado en zona. Función que permite la ubicación de fragmentos basados en reglas para mejorar la disponibilidad de cuadrículas al colocar fragmentos en diversos centros de datos, ya sea en plantas diferentes o incluso en edificios o geografías distintas.

SQL. Siglas de Structured Query Language. Véase lenguaje de consulta estructurado (SQL).

SSL. Siglas de Secure Sockets Layer. Véase capa de sockets seguros.

subclase. En Java, una clase derivada de una clase en particular, mediante herencia.

subconsulta. En SQL, subselección empleada dentro de un predicado. Por ejemplo, una sentencia select dentro de la cláusula WHERE o HAVING de otra sentencia SQL.

suceso.

1. Un cambio de estado como, por ejemplo, la finalización o la anomalía de una operación, proceso empresarial o tarea humana que puede desencadenar una acción posterior como, por ejemplo, persistir los datos del suceso en un repositorio de datos o invocar otro proceso empresarial.

2. Un cambio en los datos de una sistema de información empresarial (EIS) que es procesado por el adaptador y se utiliza para entregar objetos empresariales del EIS a los puntos finales (aplicaciones) a los que se debe notificar el cambio.

tabla de autorizaciones. Una tabla que contiene información sobre la correlación entre el rol y el usuario o grupo de usuarios y que identifica qué acceso a un recurso determinado puede tener un cliente.

TCP. Véase protocolo de control de transmisiones.

TCP/IP. Véase protocolo de control de transmisiones/protocolo Internet.

temporizador. Una tarea que produce salida en determinados momentos.

tiempo de construcción. Tiempo durante el que un programa se construye en un programa ejecutable.

tiempo de ejecución. Tiempo durante el que se está ejecutando un programa informático.

tiempo de espera. Un intervalo de tiempo asignado para que se produzca o concluya un suceso antes de interrumpir el funcionamiento.

tiempo de vida. El intervalo de tiempo en segundos que una entrada puede existir en la memoria caché antes de que se descarte.

tipo.

1. En programación Java, una clase o interfaz.

2. En un documento WSDL, elemento que contiene definiciones de tipo de datos que utiliza algún sistema de tipos (como puede ser XSD).

tipo primitivo. En Java, categoría de tipo de datos que describe una variable que contiene un solo valor del tamaño y formato pertinentes al tipo: un número, un carácter o un valor booleano. Ejemplos de tipos primitivos son byte, short, int, long, float, double, char, boolean.

Tivoli Performance Viewer. Un cliente Java que recupera los datos PMI (Performance Monitoring Infrastructure) de un servidor de aplicaciones y los muestra en varios formatos.

topología. La correlación física o lógica de la ubicación de los componentes o nodos de red dentro de una red. La topología de red más comunes son en bus, en anillo, en estrella y en árbol.

topología de despliegue. La configuración de servidores y clústeres en un entorno de despliegue y las relaciones físicas y lógicas que hay entre ellos.

topología de tiempo de ejecución . Descripción del estado momentáneo del entorno.

transacción. Proceso en el que todas las modificaciones de datos realizadas durante una transacción se confirman a la vez como una unidad o se retrotraen como un unidad.

transacción gestionada por bean (BMT) . La capacidad del bean de sesión, servlet o componente de cliente de aplicaciones de gestionar directamente sus propias transacciones, en lugar de hacerlo a través de un contenedor.

transacción global. Una unidad de trabajo recuperable efectuada por uno o varios gestores de recursos en un entorno de transacción distribuida y coordinada por un gestor de transacciones externo.

UDDI. Véase Universal Description, Discovery, and Integration.

umbral. Valor que se aplica a una interrupción dentro de una simulación, que define cuándo una simulación de proceso debe detenerse basándose en una condición existente para una proporción determinada de casos de algún suceso.

unidad de compilación. Una parte de un programa informático suficientemente completa como para ser construido correctamente.

unión.

1. Un elemento de proceso que recombina y sincroniza las vías de acceso de proceso en paralelo después de una decisión o bifurcación. Una unión espera que le lleguen datos de entrada a cada una de sus ramas entrantes antes de permitir que continúe el proceso.

2. Una operación relacional SQL en la que se pueden recuperar los datos de dos tablas, normalmente se basan en una condición de unión que especifica columnas de unión.

3. La configuración de un enlace de entrada que determina el comportamiento del enlace.

Universal Description, Discovery, and Integration (UDDI) . Conjunto de especificaciones basadas en estándares que permite que las compañías y aplicaciones encuentren y utilicen servicios web a través de Internet de forma rápida y fácil.

Universally Unique Identifier (UUID) . Identificador numérico de 128 bits que se utiliza para garantizar que dos componentes no tengan el mismo identificador.

UNIX System Services. Un elemento de z/OS que crea un entorno UNIX que cumple las especificaciones XPG4 UNIX 1995 y proporciona dos interfaces de sistema abierto en el sistema operativo z/OS: una interfaz de programación de aplicaciones (API) y una interfaz de shell interactiva.

URI. Véase Identificador universal de recursos.

URL. Siglas de Uniform Resource Locator. Véase localizador universal de recursos.

URN. Siglas de Uniform Resource Name. Véase nombre uniforme de recursos.

usuario autenticado. Un usuario de portal que ha iniciado sesión en el mismo con una cuenta válida (ID de usuario y contraseña). Los usuarios autenticados tienen acceso a todos los espacios públicos.

UUID. Véase Universally Unique Identifier.

variable. Una representación de un valor variable.

variable de entorno. Una variable que especifica cómo se ejecuta un sistema operativo u otro programa, o los dispositivos que reconoce el sistema operativo.

variable global. Una variable que se utiliza para conservar y manipular valores asignados a la misma durante la conversión y que son compartidos por correlaciones y conversiones de documentos. Uno de los tres tipos de variables soportado por el lenguaje de mandatos de correlación de Data Interchange Services.

version. Un programa bajo licencia independiente que normalmente tiene un código nuevo o una nueva función significativo.

vía de acceso de clases. Lista de directorios y archivos JAR que contienen archivos de recursos o clases Java que un programa puede cargar dinámicamente en tiempo de ejecución.

vía de acceso de construcción. La vía de acceso que se utiliza durante la compilación de código fuente Java para buscar las clases referidas que residen en otros proyectos.

virtualización. Una técnica que encapsula las características de los recursos a partir de la forma en que los demás sistemas interactúan con esos recursos.

WAR. Véase archivador web.

WCCM. Véase WebSphere Common Configuration Model.

Web Archive (WAR). Formato de archivo comprimido, definido por el estándar Java EE, para almacenar todos los recursos necesarios para instalar y ejecutar una aplicación web en un solo archivo. Véase también archivador empresarial.

WebSphere. Nombre de marca IBM que abarca las herramientas para desarrollar las aplicaciones e-business y el middleware para ejecutar las aplicaciones web.

WebSphere Common Configuration Model (WCCM) . Un modelo que proporciona acceso programático a datos de configuración.

WLM. Véase Gestor de cargas de trabajo.

WYSIWYG. Siglas de What You See Is What You Get. Véase lo que se ve es lo que se obtiene (WYSIWYG).

XA. Una interfaz bidireccional entre uno o más gestores de recursos que proporcionan acceso a recursos compartidos y un gestor de transacciones que supervisa y resuelve transacciones.

XML. Véase lenguaje de códigos ampliable.

X/Open XA. La interfaz XA de proceso de transacciones distribuidas de X/Open. Un estándar propuesto para la comunicación de transacciones distribuidas. Este estándar especifica una interfaz bidireccional entre gestores de recursos que proporcionan acceso a recursos compartidos dentro de transacciones y entre un servicio de transacciones que supervisa y resuelve transacciones.

zona desmilitarizada (DMZ) . Una configuración que incluye varios cortafuegos para añadir una capa de protección entre una intranet de la empresa y una red pública como, por ejemplo, Internet.

z/OS. Un sistema operativo de sistema principal de IBM que utiliza almacenamiento real de 64 bits.

Avisos

Las referencias en esta publicación a productos, programas o servicios de IBM no implica que IBM tenga previsto ponerlos a la venta en todos los países en los que IBM opera. Cualquier referencia a un producto, programa o servicio de IBM no pretende indicar ni implica que sólo se pueda utilizar este producto, programa o servicio de IBM. En su lugar, se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ningún derecho de propiedad intelectual de IBM. La evaluación y la verificación del funcionamiento con otros productos, excepto aquellos expresamente designados por IBM, es responsabilidad del usuario.

IBM puede tener patentes o solicitudes de patentes pendientes que conciernan al tema de este documento. La posesión de este documento no le da ninguna licencia sobre estas patentes. Puede enviar preguntas acerca de licencias por escrito a:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York 10594 Estados Unidos

Los propietarios de licencias de este programa que deseen obtener información sobre el mismo con el fin de habilitar: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido este) y (ii) el uso mutuo de la información intercambiada, se deben poner en contacto con:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
Estados Unidos
Attention: Information Requests

Esta información puede estar disponible, bajo las condiciones y los términos adecuados, incluyendo en algunos casos, el pago de una cuota.

Marcas registradas

Los siguientes términos son marcas registradas de IBM Corporation en Estados Unidos yen otros países.

- AIX
- CICS
- Cloudscape
- DB2
- Domino
- IBM
- Lotus
- RACF
- Redbooks
- Tivoli
- WebSphere
- z/OS

Java y todas las marcas registradas basadas en Java son marcas registradas de Sun Microsystems, Inc. en Estados Unidos y/o en otros países.

LINUX es una marca registrada de Linus Torvalds en Estados Unidos y/o en otros países.

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en Estados Unidos y/o en otros países.

UNIX es una marca registrada de The Open Group en Estados Unidos y en otros países.

Otros nombres de compañías, productos y servicios pueden ser marcas registradas o de servicio de terceros.

Índice

V

W

X