WebSphere® eXtreme Scale Version 7.1
Guide d'administration

WebSphere eXtreme Scale - Guide d'administration

IBM

août 2010

LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. Les informations qui y sont fournies sont susceptibles d'être modifiées avant que les produits décrits ne deviennent eux-mêmes disponibles. En outre, il peut contenir des informations ou des références concernant certains produits, logiciels ou services non annoncés dans ce pays. Cela ne signifie cependant pas qu'ils y seront annoncés.

Pour plus de détails, pour toute demande d'ordre technique, ou pour obtenir des exemplaires de documents IBM, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial.

Vous pouvez également consulter les serveurs Internet suivants :

- http://www.fr.ibm.com (serveur IBM en France)
- http://www.can.ibm.com (serveur IBM au Canada)
- http://www.ibm.com (serveur IBM aux Etats-Unis)

Compagnie IBM France
Direction Qualité
17, avenue de l'Europe
92275 Bois-Colombes Cedex

Table des matières

Figures

Tableaux

A propos du *Guide d'administration*

La documentation de WebSphere eXtreme Scale inclut trois volumes qui fournissent les informations nécessaires pour utiliser, programmer et administrer le produit WebSphere eXtreme Scale.

Bibliothèque WebSphere eXtreme Scale

La bibliothèque WebSphere eXtreme Scale contient les documents suivants :

- Le *Guide d'administration* contient les informations nécessaires pour les administrateurs système et explique notamment comment planifier les déploiements d'application, planifier la capacité, installer et configurer le produit, démarrer et arrêter des serveurs, surveiller l'environnement et le sécuriser.
- Le *Guide de programmation* contient des informations destinées aux développeurs d'applications, sur la manière de développer des applications pour WebSphere eXtreme Scale à l'aide des informations d'API incluses.
- La *Présentation du produit* contient une vue de haut niveau des concepts de WebSphere eXtreme Scale, avec des scénarios d'utilisation et des tutoriels.

Pour télécharger les documents, accédez à la page de la bibliothèque de WebSphere eXtreme Scale.

Vous pouvez également accéder à ces informations dans le Centre de documentation de WebSphere eXtreme Scale.

A qui s'adresse ce document

Ce document est principalement destiné aux administrateurs système, administrateurs de la sécurité et opérateurs système.

Structure de ce document

Ce document contient des informations sur les rubriques principales suivantes :

- Le **Chapitre 1** inclut des informations sur la mise en route.
- Le **Chapitre 2** inclut des informations sur la planification du déploiement d'application.
- Le **Chapitre 3** inclut des informations sur la planification de la capacité.
- Le **Chapitre 4** inclut des informations sur l'installation du produit.
- Le **Chapitre 5** inclut des informations sur la personnalisation de la plateforme z/OS.
- Le **Chapitre 6** inclut des informations sur la configuration du produit.
- Le **Chapitre 7** inclut des informations sur l'administration de l'environnement.
- Le **Chapitre 8** inclut des informations sur la surveillance de votre environnement de déploiement.
- Le **Chapitre 9** inclut des informations sur la sécurisation de l'environnement de déploiement.
- Le **Chapitre 10** inclut des informations sur l'identification et la résolution des problèmes de l'environnement.

- Le **Chapitre 11** inclut des informations sur le glossaire du produit.

Obtention des mises à jour de ce document

Vous pouvez obtenir les mises à jour de ce document en téléchargeant la version la plus récente à partir de la page de la bibliothèque de WebSphere eXtreme Scale.

Comment envoyer vos commentaires

Contactez l'équipe chargée de la documentation. Avez-vous trouvé ce que vous recherchiez ? Ces informations étaient-elles précises et complètes ? Envoyez vos commentaires sur cette documentation par courrier électronique, à l'adresse wasdoc@us.ibm.com.

Chapitre 1. Initiation à WebSphere eXtreme Scale

Après avoir installé WebSphere eXtreme Scale dans un environnement autonome, utilisez les étapes suivantes pour vous initier à ses fonctions en tant que grille de données en mémoire.

L'installation autonome de WebSphere eXtreme Scale comprend un exemple que vous pouvez utiliser pour vérifier votre installation et pour constater comment une grille et un client eXtreme Scale peuvent être utilisés. L'exemple de l'initiation se trouve dans le répertoire *racine_installation*/ObjectGrid/gettingstarted.

Cet exemple offre une introduction rapide aux fonctionnalités et aux opérations de base de eXtreme Scale. L'exemple contient un interpréteur de commandes et des scripts par lots conçus pour commencer une simple grille sans efforts de personnalisation. En outre, un programme client contenant la source est fourni pour exécuter les fonctions CRUD (Create, Read, Update, Delete) dans cette grille de base.

Scripts et fonctions correspondantes

Cet exemple offre les quatre scripts suivants :

le script env.sh|bat est appelé par les autres scripts pour la définition de variables d'environnement requises. Il n'est normalement pas nécessaire de modifier ce script.

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

Le script runcat.sh|bat démarre le service de catalogue eXtreme Scale sur le système local.

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

Le script runcontainer.sh|bat démarre le serveur de conteneur. Vous pouvez exécuter ce script plusieurs fois en spécifiant des noms de serveurs uniques pour démarrer autant de conteneurs que vous le voulez. Ces instances peuvent interagir pour héberger des informations partitionnées et redondantes dans la grille.

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

Le script runclient.sh|bat exécute le client CRUD et démarre l'opération voulue.

- UNIX Linux ./runclient.sh *commande valeur1 valeur2*
- Windows runclient.sh *commande valeur1 valeur2*

Pour *commande*, utilisez l'une des options suivantes :

- Spécifiez i pour insérer *valeur2* dans la grille avec la clé *valeur1*
- Spécifiez u pour mettre à jour l'objet indexé par *valeur1* avec *value2*
- Spécifiez d pour supprimer l'objet indexé par *valeur1*

- Spécifiez `g` pour extraire et afficher l'objet indexé par *valeur1*

Remarque : Le fichier*racine_installation*`/ObjectGrid/ gettingstarted/src/ Client.java` est le programme client qui indique comment établir la connexion au serveur de catalogues, obtenir une instance ObjectGrid et utiliser l'API ObjectMap.

Etapes de base

Utilisez les étapes suivantes pour démarrer votre première grille et exécuter un client en vue d'interagir avec la grille.

1. Ouvrez une session terminal ou une fenêtre de ligne de commande.
2. La commande suivante permet d'accéder au répertoire `gettingstarted` :

   `cd` *racine_installation*`/ObjectGrid/gettingstarted`

   Remplacez *racine_installation* par le chemin d'accès au répertoire racine d'installation d'eXtreme Scale ou le chemin d'accès au fichier racine du répertoire d'essai *racine_installation* d'eXtreme Scale.
3. Exécutez le script suivant pour démarrer un processus de service de catalogue sur le système hôte local :
   - UNIX Linux `./runcat.sh`
   - Windows `runcat.bat`

   Le processus du service de catalogue s'exécute dans la fenêtre du terminal en cours.
4. Ouvrez une autre session terminal ou fenêtre de ligne de commande et exécutez la commande suivante pour démarrer une instance de serveur de conteneur :
   - UNIX Linux `./runcontainer.sh server0`
   - Windows `runcontainer.bat server0`

   Le serveur de conteneur s'exécute dans la fenêtre du terminal en cours. Vous pouvez répéter les étapes 5 et 6 si vous voulez démarrer plus d'instances de serveurs de conteneur pour prendre en charge la réplication.
5. Ouvrez une autre session terminal ou fenêtre de ligne de commande pour exécuter le client.
   - Ajoutez des données à la grille :
     - UNIX Linux `./runclient.sh i key1 helloWorld`
     - Windows `runclient.bat i key1 helloWorld`
   - Recherchez et affichez la valeur :
     - UNIX Linux `./runclient.sh g key1`
     - Windows `runclient.bat g key1`
   - Mettez la valeur à jour :
     - UNIX Linux `./runclient.sh u key1 goodbyeWorld`
     - Windows `runclient.bat u key1 goodbyeWorld`
   - Supprimez la valeur :
     - UNIX Linux `./runclient.sh d key1`
     - Windows `runclient.bat d key1`
6. Arrêtez le processus de service de catalogue et les serveurs de conteneur dans les fenêtres respectives en appuyant sur <ctrl+c>.

Définition d'une grille d'objets

L'exemple utilise les fichiers `objectgrid.xml` et `deployment.xml` disponibles dans le répertoire *racine_installation*`/ObjectGrid/gettingstarted/xml` pour démarrer un serveur de conteneur. Le fichier `objectgrid.xml` est le fichier XML du descripteur d'ObjectGrid et le fichier `deployment.xml` est le fichier XML du descripteur de la règle de déploiement de la grille d'objets. Les deux fichiers réunis définissent une topologie ObjectGrid répartie.

Fichier XML du descripteur d'ObjectGrid

Un fichier XML de descripteur d'ObjectGrid permet de définir la structure de la grille d'objets utilisée par l'application. Il contient une liste de configurations de mappes de sauvegarde. Ces mappes de sauvegarde constituent l'emplacement de stockage effectif des données en cache. L'exemple suivant présente un fichier d'exemple `objectgrid.xml`. Les premières lignes de ce fichier incluent l'en-tête requis de chaque fichier XML ObjectGrid. Ce fichier d'exemple définit la grille ObjectGrid avec les mappes de sauvegarde Map1 et 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>
```

Fichier XML du descripteur de la règle de déploiement

Un fichier XML de descripteur de règle de déploiement est transmis à un serveur conteneur ObjectGrid lors du démarrage. La règle de déploiement doit être utilisée avec un fichier XML d'ObjectGrid et doit être compatible avec le fichier XML d'ObjectGrid qui lui est associé. Chaque élément objectgridDeployment de la règle de déploiement doit correspondre à un élément ObjectGrid de votre fichier XML d'ObjectGrid. Les éléments de la mappe de sauvegarde définis dans l'élément objectgridDeployment doivent être cohérents avec les mappes de sauvegarde contenues dans le fichier XML d'ObjectGrid. Chaque mappe de sauvegarde doit être référencée dans un seul et unique groupe de mappes.

Le fichier XML de descripteur de règle de déploiement est conçu pour être couplé avec le fichier XML d'ObjectGrid correspondant, le fichier `objectgrid.xml`. Dans l'exemple suivant, les premières lignes du fichier `deployment.xml` incluent l'en-tête requis de chaque fichier XML de règle de déploiement. Le fichier définit l'élément objectgridDeployment pour la grille ObjectGrid définie dans le fichier `objectgrid.xml`. Les mappes de sauvegarde Map1 et Map2 définies dans la grille ObjectGrid sont incluses dans le groupe de mappes mapSet pour lequel les attributs numberOfPartitions, minSyncReplicas et maxSyncReplicas sont configurés.

```
<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>
```

L'attribut numberOfPartitions de l'élément mapSet indique le nombre de partitions de l'élément mapSet. Il s'agit d'un attribut facultatif et sa valeur par défaut est égale à 1. Ce nombre doit être approprié pour la capacité anticipée de la grille.

L'attribut minSyncReplicas de l'élément mapSet vise à indiquer le nombre minimal de fragments répliques synchrones de chaque partition du groupe de mappes. Il s'agit d'un attribut facultatif et sa valeur par défaut est égale à 0. Le fragment primaire et le fragment réplique ne sont pas positionnés tant que le domaine ne peut pas prendre en charge le nombre minimal de fragments répliques synchrones. Pour prendre en charge la valeur minSyncReplicas, vous avez besoin d'un nombre de conteneurs égal à la valeur de minSyncReplicas plus un. Si le nombre de fragments répliques synchrones est inférieur à la valeur de minSyncReplicas, les transactions d'écrire ne sont plus autorisées pour cette partition.

L'attribut maxSyncReplicas de l'élément mapSet vise à indiquer le nombre maximal de fragments répliques synchrones de chaque partition du groupe de mappes. Il s'agit d'un attribut facultatif et sa valeur par défaut est égale à 0. Aucune autre réplique synchrone n'est placée pour une partition une fois qu'un domaine a atteint ce nombre de fragments répliques synchrones pour cette partition spécifique. L'ajout de conteneurs prenant en charge cette grille d'objets peut entraîner un nombre croissant de fragments répliques synchrones si la valeur maxSyncReplicas n'a pas déjà été atteinte. L'exemple définit la valeur maxSyncReplicas sur 1, ce qui signifie que le domaine place au maximum une réplique synchrone. Si vous démarrez plusieurs instances de serveurs de conteneur, seule une réplique synchrone sera placée dans une des instances de serveurs de conteneur.

Utilisation de la grille d'objets

Le fichier `Client.java` dans le répertoire *racine_installation*`/ObjectGrid/gettingstarted/src/` est le programme client qui illustre comment établir la connexion au serveur de catalogues, obtenir l'instance ObjectGrid et utiliser l'API ObjectMap.

Du point de vue d'une application client, l'utilisation de WebSphere eXtreme Scale englobe les étapes suivantes.

1. Connexion au service de catalogue via une instance ClientClusterContext.
2. Obtention d'une instance client ObjectGrid.
3. Obtention d'une instance Session.
4. Obtention d'une instance ObjectMap.
5. Utilisation des méthodes ObjectMap.

1. **Connexion au service de catalogue via une instance ClientClusterContext**

   Pour établir la connexion au serveur de catalogues, utilisez la méthode connect de l'API ObjectGridManager. La méthode connect utilisée nécessite uniquement le noeud final du serveur de catalogues dans le format `nomhôte:port`, tel que `localhost:2809`. Si la connexion au serveur de catalogues aboutit, la méthode connect renvoie une instance ClientClusterContext. L'instance ClientClusterContext est requise pour obtenir l'ObjectGrid à partir de l'API

ObjectGridManager. Le fragment de code suivant montre comment établir la connexion à un serveur de catalogues et comment obtenir une instance ClientClusterContext.

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

2. **Obtention d'une instance ObjectGrid**

   Pour obtenir une instance ObjectGrid, utilisez la méthode getObjectGrid de l'API ObjectGridManager. La méthode getObjectGrid requiert l'instance ClientClusterContext et le nom de l'instance ObjectGrid. L'instance ClientClusterContext est obtenue pendant la connexion au serveur de catalogues. Le nom de l'ObjectGrid correspond à la grille `Grid` spécifiée dans le fichier `objectgrid.xml`. Le fragment de code suivant montre comment obtenir l'ObjectGrid en appelant la méthode getObjectGrid de l'API ObjectGridManager.

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

3. **Obtention d'une instance Session**

   Vous pouvez obtenir une session de l'instance ObjectGrid obtenue. Une instance Session est requise pour obtenir l'instance ObjectMap et pour effectuer une démarcation de transaction. Le fragment de code suivant montre comment obtenir une instance Session en appelant la méthode getSession de l'API ObjectGrid.

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

4. **Obtention d'une instance ObjectMap**

   Après l'obtention d'une instance Session, vous pouvez obtenir une instance ObjectMap d'une instance Session en appelant la méthode getMap de l'API Session. Vous devez transmettre le nom de la mappe en tant que paramètre à la méthode getMap afin d'obtenir l'instance ObjectMap. Le fragment de code suivant montre comment obtenir l'instance ObjectMap en appelant la méthode getMap de l'API Session.

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

5. **Utilisation des méthodes ObjectMap**

   Après l'obtention d'une instance ObjectMap, vous pouvez utiliser l'API ObjectMap. Gardez à l'esprit que l'interface ObjectMap est une mappe transactionnelle et requiert une démarcation de transaction à l'aide des méthodes begin et commit de l'API Session. Si aucune démarcation de transaction explicite n'a lieu dans l'application, les opérations ObjectMap s'exécutent sans transactions de validation automatique.

   Le fragment de code suivant montre comment utiliser l'API ObjectMap avec des transactions de validation automatique.

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

   Le fragment de code suivant montre comment utiliser l'API ObjectMap avec une démarcation de transaction explicite.

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

Informations supplémentaires

Cet exemple illustre comment démarrer le serveur de catalogues et le serveur conteneur et comment utiliser l'API ObjectMap dans un environnement autonome. Vous pouvez également faire appel à l'API EntityManager.

Dans un environnement WebSphere Application Server dans lequel WebSphere eXtreme Scale est installé ou activé, une topologie de connexion en réseau constitue

le scénario le plus fréquent. Dans une topologie de connexion en réseau, le serveur de catalogues est hébergé dans le processus du gestionnaire de déploiement WebSphere Application Server et chaque instance WebSphere Application Server héberge automatiquement un serveur eXtreme Scale. Pour que la grille d'objets soit disponible automatiquement, il suffit que les applications Java™ Platform, Enterprise Edition contiennent le fichier XML du descripteur d'ObjectGrid et le fichier XML du descripteur de règle de déploiement ObjectGrid dans le répertoire `META-INF`. Une application peut ensuite se connecter à un serveur de catalogues disponible en local et obtenir une instance ObjectGrid.

Conventions concernant les répertoires

A l'aide de nombreux exemples et en exposant la syntaxe de ligne de commande, nous allons expliquer comment faire référence à des répertoires spéciaux comme *racine_install_wxs*, et *rép_base_wxs*. Ces répertoires sont définis comme suit.

racine_install_wxs
Le répertoire *racine_install_wxs* est le répertoire racine où sont installés les fichiers WebSphere eXtreme Scale. Il peut s'agir du répertoire dans lequel est extrait le fichier zip de la version d'essai ou de celui dans lequel est installé le produit eXtreme Scale.

- exemple où la version d'essai a été extraite :
  `/opt/IBM/WebSphere/eXtremeScale`
- example où eXtreme Scale est installé dans un répertoire autonome :
  `/opt/IBM/eXtremeScale`
- example où eXtreme Scale est intégré à WebSphere Application Server :
  `/opt/IBM/WebSphere/AppServer`

rép_base_wxs
Le répertoire *rép_base_wxs* est le répertoire racine des bibliothèques, des exemples et des composants du produit WebSphere eXtreme Scale. Ce répertoire est le même que *racine_install_wxs* lorsque la version d'essai est extraite. Pour les installations autonomes, il s'agit du sous-répertoire ObjectGrid du répertoire *racine_install_wxs*. Pour les installations intégrées à WebSphere Application Server, ce répertoire est le répertoire `optionalLibraries/ObjectGrid` du répertoire *racine_install_wxs*.

- exemple où la version d'essai a été extraite :
  `/opt/IBM/WebSphere/eXtremeScale`
- example où eXtreme Scale est installé dans un répertoire autonome :
  `/opt/IBM/eXtremeScale/ObjectGrid`
- example où eXtreme Scale est intégré à WebSphere Application Server :
  `/opt/IBM/WebSphere/AppServer/optionalLibraries/ObjectGrid`

racine_was
Le répertoire *racine_was* est le répertoire racine d'une installation WebSphere Application Server :

`/opt/IBM/WebSphere/AppServer`

rép_base_servicerest
Le répertoire *rép_base_servicerest* est le répertoire dans lequel se trouvent les bibliothèques et les exemples du service de données REST d'eXtreme Scale. Ce répertoire porte le nom de *restservice* et c'est un sous-répertoire du répertoire *rép_base_wxs*.

- exemple pour les déploiements autonomes :

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

- exemple pour les déploiements intégrés à WebSphere Application Server :

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

racine_tomcat
Le répertoire *racine_tomcat* est le répertoire racine de l'installation d'Apache Tomcat.

/opt/tomcat5.5

racine_wasce
Le répertoire *racine_wasce* est le répertoire racine de l'installation de WebSphere Application Server Community Edition.

/opt/IBM/WebSphere/AppServerCE

rép_base_java
Le répertoire *rép_base_java* est le répertoire racine d'une installation de Java Runtime Environment Kit (JRE).

/opt/IBM/WebSphere/eXtremeScale/java

Chapitre 2. Planification de la capacité

Si la taille initiale et la taille projetée des données ont été définies, vous pouvez planifier la capacité dont vous avez besoin pour exécuter WebSphere eXtreme Scale. Bien que cette planification vous aide à déployer eXtreme Scale de manière efficace lors de modifications futures, elle vous permet d'optimiser l'élasticité d'eXtreme Scale, dont vous ne pourriez pas bénéficier dans un autre scénario, par exemple avec une base de données en mémoire ou un autre type de base de données.

Extensibilité

L'extensibilité est ce qui permet aux données d'un déploiement de WebSphere eXtreme Scale d'être réparties dans un ensemble de serveurs (conteneurs) en fonction des choix de configurations que vous avez opérés.

Définition de la taille de la mémoire et calcul du nombre de partitions

Vous pouvez calculer la quantité de mémoire et le nombre de partitions nécessaires pour votre configuration.

WebSphere eXtreme Scale stocke les données dans l'espace adresse de machines virtuelles Java (JVM). Chaque JVM fournit un espace processeur pour traiter la création, la récupération, la mise à jour et la suppression d'appels pour les données stockées dans la JVM. En outre, chaque JVM fournit de l'espace mémoire pour les serveurs secondaires et les entrées de données. Les objets Java varient en taille. Par conséquent, vous devez effectuer une mesure afin d'estimer la quantité de mémoire nécessaire.

Pour adapter la taille de la mémoire à vos besoins, chargez les données d'application dans une seule JVM. Lorsque l'utilisation de segment de mémoire atteint 60 %, notez le nombre d'objets utilisés. Ce nombre correspond au nombre d'objets maximal recommandé pour chaque machines virtuelles Java. Pour obtenir la définition de taille la mieux adaptée, utilisez des données réalistes et introduisez tout index défini, car les index occupent également de la mémoire. Le meilleur moyen de mesurer l'utilisation de la mémoire est d'exécuter une sortie verbosegc de récupération de place, car elle vous donne le résultat après l'opération de récupération de place. Vous pouvez demander à tout moment via les beans gérés ou à l'aide d'un programme à connaître l'utilisation d'un segment de mémoire, mais ces requêtes ne vous donnent qu'un cliché instantané du segment de mémoire, susceptible de contenir des éléments inutiles non supprimés. C'est pour cela que cette méthode ne constitue pas un moyen précis de connaître la mémoire utilisée.

Mise à l'échelle de la configuration

Nombre de fragments par partition (valeur numShardsPerPartition)

Pour calculer le nombre de fragments par partition, ou valeur numShardsPerPartition, ajoutez 1 pour le fragment primaire plus le nombre total de fragments répliques souhaité.

```
numShardsPerPartition = 1 + total_number_of_replicas
```

Nombre de machines virtuelles Java (valeur minNumJVMs)

Pour mettre à l'échelle votre configuration, décidez d'abord du nombre total maximal d'objets à stocker. Pour déterminer le nombre de machines virtuelles Java nécessaire, utilisez la formule suivante :

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

Arrondissez cette valeur à l'entier le plus près.

Nombre de fragments (valeur numShards)

Pour la taille finale, 10 fragments doivent être utilisés pour chaque JVM. Comme décrit ci-dessus, chaque JVM présente un fragment primaire et (N-1) fragments répliques, soit dans ce cas, 9 fragments répliques. Etant donné que vous disposez déjà du nombre de machines virtuelles Java pour le stockage de données, vous pouvez multiplier le nombre de machines virtuelles Java par 10 pour obtenir le nombre de fragments :

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

Nombre de partitions

Si une partition dispose déjà d'un fragment primaire et d'un fragment réplique, cette partition a donc deux fragments (le primaire et le réplique). Le nombre de partitions correspond au nombre de fragments divisé par 2 et arrondi au nombre premier le plus proche. Si la partition présente un fragment primaire et deux secondaires, le nombre de partitions correspond au nombre de fragments divisé par 3 et arrondi au nombre premier le plus proche.

```
numPartitions = numShards / numShardsPerPartition
```

Exemple de mise à l'échelle

Dans cet exemple, le nombre d'entrées commence à 250 millions. Chaque année, le nombre d'entrées croît d'environ 14 %. En sept jours, le nombre total d'entrées devient 500 millions. Vous devez planifier la capacité en conséquence. Pour une haute disponibilité, un serveur secondaire est nécessaire. Avec un serveur secondaire, le nombre d'entrées double et devient 1 milliards d'entrées. Dans le cadre d'un test, deux millions d'entrées peuvent être stockées dans chaque JVM. L'utilisation de calculs pour ce scénario montre le besoin de la configuration suivante :

- 500 machines virtuelles Java pour stocker le nombre final d'entrées.
- 5 000 fragments, obtenus en multipliant 500 machines virtuelles Java par 10.
- 2 500 partitions, arrondies à 2503 (nombre premier supérieur le plus proche), calculées en prenant 5 000 fragments, divisés par deux pour les fragments primaires et secondaires.

Début de la configuration

En s'appuyant sur les calculs précédents, vous commenceriez avec 250 machines virtuelles Java, quantité qui augmenterait jusqu'à 500 machines virtuelles Java en cinq ans et vous permettrait de gérer une croissance incrémentielle jusqu'à atteindre le nombre d'entrées final.

Dans cette configuration, environ 200 000 entrées sont stockées par partition (500 millions d'entrées divisées par 2503 partitions). Vous devez définir le paramètre

numberOfBuckets sur la mappe qui contient les entrées sur le nombre premier supérieur le plus proche. Dans cet exemple 70 887, qui maintient le rapport autour de 3.

Le nombre maximal de machines virtuelles Java est atteint

Lorsque vous atteignez le nombre maximal de 500 machines virtuelles Java, vous pouvez toujours accroître votre grille. Lorsque le nombre de machines virtuelles Java dépasse le nombre maximal de 500, le nombre de fragments commence à tomber en dessous de 10 pour chaque JVM, ce qui est inférieur au nombre recommandé. La taille des fragments augmente et risque d'entraîner des problèmes. Vous devez répéter le processus de définition de la taille en fonction de la croissance future et redéfinir le nombre de partitions. Cette pratique requiert un redémarrage complet de la grille ou sa mise en indisponibilité.

Nombre de serveurs

Avertissement : N'utilisez la pagination sur un serveur sous aucune circonstance.

Une seule JVM utilise plus de mémoire que la taille d'un segment de mémoire. Par exemple, avec 1 Go de segment de mémoire, une JVM utilise en fait 1,4 Go de mémoire réelle. Déterminez la mémoire vive disponible sur le serveur. Divisez la quantité de mémoire vive par la quantité de mémoire pour chaque JVM pour obtenir le nombre maximal de machines virtuelles Java sur le serveur.

Définition du nombre d'unités centrales par partition

Bien que l'une des fonctions principales d'eXtreme Scale soit sa capacité d'évolutivité, il est également important d'évaluer et d'adapter le nombre idéal d'unités centrales en vue d'une montée en charge.

Les coûts liés au processeur comprennent :

- Coût de gestion des opérations de création, d'extraction, de mise à jour et de suppression à partir des clients.
- Coût de la réplication à partir d'autres machines virtuelles Java.
- Coût de l'invalidation.
- Coût de la stratégie d'expulsion.
- Coût de la récupération de place.
- Coût de la logique d'application.

machines virtuelles Java par serveur

Utilisez deux serveurs et démarrez le nombre maximal de JVM par serveur. Utilisez le nombre de partitions calculé à la section précédente. Ensuite, préchargez dans ces machines virtuelles Java une quantité de données ne dépassant pas la capacité des deux ordinateurs. Utilisez un serveur distinct en tant que client. Exécutez une simulation de transaction réaliste sur cette grille de deux serveurs.

Pour calculer la valeur de référence, essayez de saturer l'utilisation du processeur. Si vous n'y parvenez pas, c'est probablement parce que le réseau est saturé. Dans ce cas, ajoutez des cartes réseau et procédez à une permutation circulaire de ces machines virtuelles Java.

Exécutez les ordinateurs à 60% d'utilisation du processeur et mesurez le taux de transactions de création, d'extraction, de mise à jour et de suppression. Cette mesure indique la capacité de traitement des deux serveurs. Ce nombre double avec quatre serveurs, double encore avec huit serveurs, etc. Cette progression suppose que la capacité du réseau et la capacité du client peuvent également progresser.

Les temps de réponse d'eXtreme Scale doivent donc rester stable au fur et à mesure que le nombre de serveurs évolue. La capacité de traitement des transactions doit progresser de manière linéaire au fur et à mesure que des ordinateurs sont ajoutés à la grille.

Définition de la taille d'unités centrales pour des transactions parallèles

Les transactions à partition unique présentent une mise à l'échelle linéaire du débit par rapport à l'augmentation de la taille de la grille. Les transactions parallèles diffèrent des transactions à partition unique, car elles affectent un ensemble de serveurs (cet ensemble peut comprendre tous les serveurs).

Si une transaction affecte tous les serveurs, le débit est limité au débit du client ayant initié la transaction ou au serveur affecté le plus lent. Les grilles de grande taille répartissent davantage les données et fournissent plus d'espace processeur, de mémoire, de réseau, etc. Toutefois, le client doit attendre la réponse du serveur le plus lent et doit utiliser les résultats de la transaction.

Lorsqu'une transaction affecte un sous-ensemble de serveurs, M sur N serveurs reçoivent une requête. Le débit est alors "N divisé par M" fois plus vite que le débit du serveur le plus lent. Par exemple, si vous disposez de 20 serveurs et d'une transaction qui affecte 5 serveurs, le débit est 4 fois supérieur au débit du serveur le plus lent de la grille.

Lorsqu'une transaction parallèle se termine, les résultats sont envoyés à l'unité d'exécution du client ayant commencé la transaction. Ce client doit ensuite procéder à l'agrégation des résultats en unité d'exécution simple. Le temps d'agrégation augmente avec l'augmentation du nombre de serveurs affectés par la transaction. Toutefois, cette durée dépend de l'application, car il se peut que chaque serveur renvoie un résultat plus petit au fur et à mesure que la grille augmente.

Généralement, les transactions parallèles affectent tous les serveurs de la grille, car les partitions sont réparties uniformément dans la grille. Dans ce cas, le débit se limite à la première hypothèse.

Récapitulatif

Avec cette définition de taille, vous disposez de trois métriques, comme suit.

- Nombre de partitions.
- Nombre de serveurs nécessaires pour la mémoire requise.
- Nombre de serveurs nécessaires pour le débit requis.

Si vous avez besoin de 10 serveurs pour la quantité de mémoire nécessaire, mais que vous obtenez uniquement 50 % du débit requis en raison d'une saturation du processeur, vous devait avoir deux fois plus de serveurs.

Pour une stabilité optimale, vous devez exécuter les serveurs à 60 % du chargement de processeur et les segments de mémoire de JVM à 60 % du chargement des segments de mémoire. Les renforts peuvent ensuite pousser l'utilisation du processeur à 80-90 %, mais n'exécutent pas régulièrement vos serveurs à un niveau supérieur à ces niveaux.

Planification de la capacité et haute disponibilité (mise en cache dynamique)

L'API de cache dynamique est disponible pour les applications Java EE qui sont déployées dans WebSphere Application Server. Ce cache dynamique peut être optimisé pour mettre en cache les données métier et les fichiers HTML générés ou pour synchroniser les données en cache de la cellule à l'aide du service DRS de réplication des données.

Présentation

Toutes les instances du cache dynamique créées avec le fournisseur de cache dynamique de WebSphere eXtreme Scale sont par défaut à disponibilité élevée. Le niveau et le coût de la mémoire de la haute disponibilité dépendent de la topologie utilisée.

Si vous utilisez une topologie imbriquée, la taille du cache est limitée à la quantité de mémoire disponible dans un processus serveur unique et chaque processus serveur stocke une copie complète du cache. Tant que l'exécution de ce processus serveur unique se poursuit, le cache survit. Les données de cette mémoire sont perdues uniquement en cas d'arrêt de tous les serveurs qui y accèdent.

Dans le cas d'une topologie partitionnée imbriquée, la taille du cache est limitée à l'ensemble de l'espace disponible dans tous les processus serveur. Le fournisseur de cache dynamique eXtreme Scale par défaut utilise une réplique pour chaque fragment primaire, de sorte que chaque donnée mise en cache est stockée deux fois.

Utilisez la formule A pour déterminer la capacité du cache partitionné imbriqué.

Formule A

D * C / (1 + S) = M

Où :

- D = Mémoire disponible par processus conteneur
- C = nombre de conteneurs
- S = nombre de fragments répliques
- M = taille totale du cache

Prenons le cas d'une grille WebSphere Network Deployment dans laquelle chaque processus dispose de 256 Mo d'espace disponible et qui compte un total de 4 processus serveur : une instance du cache de tous ces serveurs peut stocker jusqu'à 512 mégaoctets de données. Dans ce mode, le cache peut survivre à une panne de serveur sans perte de données. De la même manière, deux serveurs au maximum peuvent être arrêtés séquentiellement sans perte de données. Pour l'exemple suivant, la formule est la suivante :

256 Mo * 4 conteneurs/ (1 primaire + 1 réplique) = 512 Mo.

Les caractéristiques de taille des mémoires cache utilisant une topologie distante sont similaires à celles qui utilisent une topologie partitionnée imbriquée, mais les premières sont limitées à la quantité d'espace disponible dans tous les processus conteneur eXtreme Scale.

Dans une topologie distante, il est possible d'augmenter le nombre de fragments répliques pour atteindre un niveau de disponibilité plus élevé, au prix d'une augmentation de la quantité de mémoire. Dans la plupart des applications dont le cache est dynamique, cette opération n'est pas nécessaire, mais vous pouvez éditer le fichier dynacache-remote-deployment.xml pour augmenter le nombre de fragments répliques.

Utilisez les formules B et C suivantes pour déterminer l'effet de l'ajout de fragments répliques sur la haute disponibilité du cache.

Formule B

N = Minimum(T -1, S)

Où :

- N = nombre d'échecs simultanés de processus
- T = nombre total de conteneurs
- S = nombre total de fragments répliques

Formule C

Ceiling(T/ (1+N)) = m

Où :

- T = nombre total de conteneurs
- N = nombre total de fragments répliques
- m = nombre minimal de conteneurs nécessaires pour prendre en charge les données en cache.

Sur l'optimisation des performances avec le fournisseur de cache dynamique, voir Optimisation du fournisseur de cache dynamique.

Définition de la taille du cache

Avant de pouvoir déployer une application utilisant le fournisseur de cache dynamique WebSphere eXtreme Scale, vous devez combiner les principes généraux décrits à la section précédente et les données environnementales des systèmes de production. Les premiers chiffres à identifier sont le nombre total de processus conteneur et la quantité de mémoire disponible dans chaque processus pouvant contenir les données du cache. Dans une topologie imbriquée, les conteneurs du cache sont aussi localisés dans les processus serveur WebSphere Application, de sorte qu'il existe un conteneur par serveur partageant le cache. Le meilleur moyen de définir l'espace disponible dans le processus est d'identifier la quantité de mémoire supplémentaire de l'application sans activer la mise en cache et WebSphere Application Server. Pour cela, vous pouvez analyser les données détaillées de récupération de place. Lorsque la topologie utilisée est distante, vous trouvez ces informations dans la sortie détaillée de la récupération de place d'un

conteneur autonome ayant démarré récemment et dans lequel aucune donnée de cache n'a encore été entrée. Dernier élément auquel vous devez penser : vous devez réserver des segments de la mémoire pour la récupération de place. La somme de l'espace restant dans le conteneur WebSphere Application Server ou dans le conteneur autonome et de la taille réservée pour le cache ne doit pas dépasser 70 % du total des segments de mémoire.

Une fois ces informations collectées, les valeurs peuvent être entrées dans la formule A décrite précédemment pour déterminer la taille maximale du cache partitionné. Une fois cette taille connue, l'étape suivante consiste à déterminer le nombre total d'entrées du cache pouvant être prises en charge, ce qui suppose d'identifier la taille moyenne de chaque entrée. Il suffit pour cela d'ajouter 10% à la taille de l'objet client. Pour obtenir des informations plus détaillées sur la définition de la taille des entrées du cache lors d'une utilisation du cache dynamique, voir le Guide d'optimisation du cache dynamique et du service de réplication des données.

Lorsque la compression est activée, elle affecte la taille de l'objet client et non l'espace restant dans le système de mise en cache. Utilisez la formule suivante pour déterminer la taille d'un objet mis en cache lorsque la compression est utilisée :

T = O * C + O * 0.10

Où :

- T = Taille moyenne de l'objet mis en cache
- O = Taille moyenne de l'objet client non compressé
- C = Rapport de compression exprimé sous la forme d'une fraction.

Un rapport de compression de 2 à 1 est égal à 1/2 = 0.50. Une valeur moins élevée est recommandée. Si l'objet stocké est un objet Java simple normal principalement rempli de types primitifs, vous pouvez supposer que le rapport de compression est de l'ordre de 0,60 à 0,70. Si l'objet mis en cache est un objet servlet, JSP ou WebServices, la meilleure méthode pour déterminer le rapport de compression consiste à compresser un exemple représentatif avec un utilitaire de compression ZIP. Si cette opération est impossible, considérez qu'un rapport de compression compris entre 0,2 et 0,35 est fréquent pour ce type de données.

Ensuite, utilisez ces informations pour déterminer le nombre total d'entrées du cache qui peuvent être prises en charge. Utilisez la formule D suivante :

Formule D

T = S / M

Où :

- T = Nombre total d'entrées du cache
- S = Taille totale disponible pour les données du cache, calculée à l'aide de la formule A
- M = Taille moyenne de chaque entrée du cache

Enfin, vous devez définir la taille de l'instance de cache dynamique pour appliquer cette limite. A cet égard, le fournisseur de cache dynamique WebSphere eXtreme Scale est différent du fournisseur de cache dynamique par défaut. Utilisez la formule suivante pour déterminer la valeur de la taille de l'instance de cache dynamique. Cette formule est la suivante :

Formule E

Ct = Tt / Np

Où :

- Tt = Taille cible totale du cache
- Ct = Paramètre de la taille du cache à définir dans l'instance de cache dynamique
- Np = Nombre de partitions. La valeur par défaut est 47.

Associez la taille de l'instance de cache dynamique à une valeur calculée par la formule E pour chaque serveur partageant l'instance du cache.

Chapitre 3. Installation et déploiement de WebSphere eXtreme Scale

WebSphere eXtreme Scale est une grille de données en mémoire que vous pouvez utiliser pour partitionner, répliquer et gérer dynamiquement des données d'application et une logique métier entre plusieurs serveurs. Une fois que vous avez déterminé les rôles et exigences de votre déploiement, installez eXtreme Scale sur votre système.

Avant de commencer

- Etablissez la manière dont WebSphere eXtreme Scale s'intègre dans votre topologie actuelle. Pour plus d'informations, voir la présentation de l'architecture et de la topologie de WebSphere eXtreme Scale, dans le *Présentation du produit*.
- Vérifiez que votre environnement remplit les conditions requises pour installer eXtreme Scale. Pour plus d'informations, voir «Configuration matérielle et logicielle requise», à la page 63.

Pourquoi et quand exécuter cette tâche

7.1+ **Deux types d'installation**

- l'installation complète de WebSphere eXtreme Scale : elle permet d'installer le serveur, le client ou les deux
- l'installation de WebSphere eXtreme Scale Client : elle permet d'installer le client sur des plateformes spécifiques

Environnements pris en charge

Vous n'êtes pas obligé d'installer et de déployer eXtreme Scale sur un niveau de système d'exploitation spécifique. Chaque installation Java Platform, Standard Edition (J2SE) et Java Platform, Enterprise Edition (JEE) requiert des niveaux de système d'exploitation ou des correctifs différents.

Vous pouvez installer et déployer le produit dans les environnements JEE et J2SE. Vous pouvez également regrouper le composant client avec les applications JEE directement sans les intégrer à WebSphere Application Server. WebSphere eXtreme Scale prend en charge Java Runtime Environment (JRE) Version 1.4.2 et les versions ultérieures et WebSphere Application Server Version 6.0.2 et les versions ultérieures.

Procédure

- Installez WebSphere eXtreme Scale ou WebSphere eXtreme Scale Clientautonomes.

  Vous pouvez installer le produit eXtreme Scale autonome dans un environnement qui ne contient pas WebSphere Application Server ou WebSphere Application Server Network Deployment. Si vous choisissez une installation autonome, vous définissez un nouvel emplacement d'installation dans lequel le serveur eXtreme Scale sera installé.

Avertissement : Vous pouvez également utiliser un profil non root (non administrateur) pour WebSphere eXtreme Scale dans un environnement autonome. Un environnement autonome est un environnement qui n'utilise pas WebSphere Application Server. Pour utiliser un profil non root, vous devez remplacer le propriétaire du répertoire `ObjectGrid` par le profil non root. Vous pouvez alors vous connecter avec ce profil non root et utiliser eXtreme Scale comme vous le feriez avec un profil root (administrateur).

- Intégrez le produit à WebSphere Application Server ou à WebSphere Application Server Network Deployment.

  Vous pouvez installer eXtreme Scale et l'intégrer à une installation existante de WebSphere Application Server ou WebSphere Application Server Network Deployment. Avec l'installation complète, vous pouvez sélectionner d'installer aussi bien le client que le serveur eXtreme Scale ou de n'installer que le client.
- Créez et étendez des profils.

  Créez et étendez des profils pour utiliser les fonctions d'eXtreme Scale. Si vous exécutez WebSphere Application Server Version 6.1 ou 7.0, vous pouvez utiliser le plug-in de l'outil de gestion de profil ou la commande manageprofiles. Si vous exécutez WebSphere Application Server Version 6.0.2, vous devez utiliser la commande wasprofile pour créer et étendre des profils.
- Appliquez la maintenance.

  Utilisez IBM Update Installer Version 7.0.0.4 ou version ultérieure pour appliquer la maintenance à votre environnement.

Migration vers WebSphere eXtreme Scale Version 7.1

Le programme d'installation de WebSphere eXtreme Scale ne vous permet pas de mettre à niveau ou de modifier une précédente installation. Vous devez désinstaller la version précédente avant d'installer la nouvelle version. Vous n'avez pas besoin de faire migrer vos fichiers de configuration car leur compatibilité est ascendante. Mais, si vous avez modifié l'un des scripts qui sont livrés avec le produit, vous devrez réappliquer ces changements aux scripts modifiés.

Avant de commencer

Vérifiez que vos systèmes disposent de la configuration minimale requise pour les versions du produit que vous avez l'intention de migrer et d'installer. Pour plus d'informations, voir «Configuration matérielle et logicielle requise», à la page 63.

Pourquoi et quand exécuter cette tâche

Fusionnez les fichiers script modifiés du produit avec les nouveaux fichiers scripts du produit dans le répertoire `/bin` pour conserver vos modifications.

Conseil : Si vous n'avez pas modifié les fichiers script installés avec le produit, vous n'avez pas besoin d'effectuer les étapes de migration ci-après. A la place, vous pouvez migrer vers la version 7.1 en désinstallant la version précédente et en installant la nouvelle version dans le même répertoire.

Procédure

1. Arrêtez tous les processus qui utilisent eXtreme Scale.
   - Pour arrêter tous les processus exécutés dans votre environnement eXtreme Scale autonome, consultez la rubrique sur l'arrêt des serveurs autonomes.

- Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server ou WebSphere Application Server Network Deployment, reportez-vous aux utilitaires de ligne de commande.

2. Sauvegardez les scripts modifiés de votre répertoire d'installation actuel dans un répertoire temporaire.
3. Désinstallez le produit.
4. Installez eXtreme Scale Version 7.1. Pour plus d'informations, voir Chapitre 3, «Installation et déploiement de WebSphere eXtreme Scale», à la page 17.
5. Fusionnez vos modifications apportées aux fichiers du répertoire temporaire avec les nouveaux fichiers scripts du produit, dans le répertoire `/bin`.
6. Démarrez tous vos processus eXtreme Scale pour commencer à utiliser le produit. Pour plus d'informations, voir «Terminologie des tâches administratives», à la page 319 les explications sur l'administration de votre environnement.

Installation du produit WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client autonome

Vous pouvez installer le produit WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client autonome dans un environnement qui ne contient pas WebSphere Application Server ou WebSphere Application Server Network Deployment.

Avant de commencer

- Vérifiez que le répertoire d'installation cible est vide ou qu'il n'existe pas.

  Important : Si une version précédente d'WebSphere eXtreme Scale ou le composant ObjectGrid se trouve dans le répertoire que vous spécifiez pour installer la version 7.1, le produit n'est pas installé. Par exemple, il peut exister un dossier `<racine_install_wxs>/ObjectGrid`. Vous pouvez sélectionner un autre répertoire d'installation ou annuler l'installation. Ensuite, désinstallez l'installation précédente et exécutez de nouveau l'assistant.

- **7.1+** IBM Runtime Environment, Java Technology Edition, Version 6 SR4 est installé dans le cadre de l'installation autonome dans le dossier *<rép_base_install_wxs>*`/java`.

Pourquoi et quand exécuter cette tâche

Si vous installez le produit comme produit autonome, installez le client et le serveur WebSphere eXtreme Scale indépendamment. Avec l'installation de WebSphere eXtreme Scale Client en mode autonome, vous installez un client pour WebSphere eXtreme Scale. Par conséquent, les processus serveur et client accèdent à toutes les ressources requises en local. Vous pouvez également imbriquer WebSphere eXtreme Scale dans des applications Java Platform, Standard Edition (J2SE) existantes à l'aide de scripts et de fichiers d'archive Java (JAR).

Tableau 1. Fichiers d'exécution d'une installation intégrale de WebSphere eXtreme Scale. WebSphere eXtreme Scale s'appuie sur les processus ObjectGrid et les API associées. Le tableau ci-après répertorie les fichiers JAR inclus dans l'installation.

Nom de fichier	Environnement	Emplacement d'installation	Description
wxsdynacache.jar	Client et serveur	dynacache/lib	Le fichier wxsdynacache.jar contient les classes nécessaires à utiliser avec le fournisseur de mémoire cache dynamique. Le fichier est automatiquement inclus dans l'environnement d'exécution du serveur lorsque vous utilisez les scripts fournis.
wxshyperic.jar	Utilitaire	hyperic/lib	Plug-in de détection du serveur WebSphere eXtreme Scale pour l'agent de surveillance SpringSource Hyperic.
objectgrid.jar	Local, client et serveur	lib	Le fichier objectgrid.jar est utilisé par l'environnement d'exécution du serveur de J2SE Version 1.4.2 et versions ultérieures. Le fichier est automatiquement inclus dans l'environnement d'exécution du serveur lorsque vous utilisez les scripts fournis.
ogagent.jar	Local, client et serveur	lib	Le fichier ogagent.jar contient les classes d'exécution requises pour exécuter l'agent d'instrumentation Java utilisé avec l'API EntityManager.
ogclient.jar	Local et client	lib	Le fichier ogclient.jar ne contient que les environnements d'exécution local et client. Vous pouvez utiliser ce fichier avec J2SE Version 1.4.2 et versions ultérieures.
ogspring.jar	Local, client et serveur	lib	Le fichier ogspring.jar contient les classes de support de l'intégration de l'infrastructure SpringSource Spring.
wsogclient.jar	Local et client	lib	Fichier wsogclient.jar installé si vous utilisez un environnement qui contient WebSphere Application Server Version 6.0.2 et versions ultérieures. Ce fichier ne contient que les environnements d'exécution local et client.
wxssizeagent.jar	Local, client et serveur	lib	Le fichier wxssizeagent.jar permet de fournir des informations plus précises sur le dimensionnement des entrées de la mémoire cache lors de l'utilisation de l'environnement d'exécution Java (JRE) Version 1.5 ou ultérieure.
ibmcfw.jar ibmext.jar ibmorb.jar ibmorbapi.jar	Client et serveur	libendorsed	Cet ensemble de fichiers inclut l'environnement d'exécution de l'ORB (Object Request Broker) qui permet d'exécuter des applications dans des processus Java SE.
restservice.ear	Client	restservice/lib	Le fichier restservice.ear contient l'archive d'entreprise de l'application du service de données REST d'eXtreme Scale pour les environnements WebSphere Application Server.
restservice.war	Client	restservice/lib	Le fichier restservice.war contient l'archive Web du service de données REST d'eXtreme Scale pour les serveurs d'applications acquis auprès d'un autre fournisseur.
xsadmin.jar	Utilitaire	samples	Le fichier xsadmin.jar contient l'exemple d'utilitaire d'administration d'eXtreme Scale.
sessionobjectgrid.jar	Client et serveur	session/lib	Le fichier sessionobjectgrid.jar contient l'environnement d'exécution de gestion des sessions HTTP d'eXtreme Scale.
splicerlistener.jar	Utilitaire	session/lib	Le fichier splicerlistener.jar contient l'utilitaire splicer du filtre du gestionnaire des sessions HTTP d'eXtreme Scale.
xsgbean.jar	Serveur	wasce/lib	Le fichier xsgbean.jar contient le GBean permettant d'imbriquer les serveurs eXtreme Scale dans des serveurs d'applications WebSphere Application Server Community Edition.

Tableau 1. Fichiers d'exécution d'une installation intégrale de WebSphere eXtreme Scale (suite). WebSphere eXtreme Scale s'appuie sur les processus ObjectGrid et les API associées. Le tableau ci-après répertorie les fichiers JAR inclus dans l'installation.

Nom de fichier	Environnement	Emplacement d'installation	Description
splicer.jar	Utilitaire		

Tableau 2. Fichiers d'exécution de WebSphere eXtreme Scale Client. WebSphere eXtreme Scale Client s'appuie sur les processus ObjectGrid et les API associées. Le tableau ci-après répertorie les fichiers JAR inclus dans l'installation.

Nom de fichier	Environnement	Emplacement d'installation	Description
wxsdynacache.jar	Client et serveur	dynacache/lib	Le fichier wxsdynacache.jar contient les classes nécessaires à utiliser avec le fournisseur de mémoire cache dynamique. Le fichier est automatiquement inclus dans l'environnement d'exécution du serveur lorsque vous utilisez les scripts fournis.
wxshyperic.jar	Utilitaire	hyperic/lib	Plug-in de détection du serveur WebSphere eXtreme Scale pour l'agent de surveillance SpringSource Hyperic.
ogagent.jar	Local, client et serveur	lib	Le fichier ogagent.jar contient les classes d'exécution requises pour exécuter l'agent d'instrumentation Java utilisé avec l'API EntityManager.
ogclient.jar	Local et client	lib	Le fichier ogclient.jar ne contient que les environnements d'exécution local et client. Vous pouvez utiliser ce fichier avec J2SE Version 1.4.2 et versions ultérieures.
ogspring.jar	Local, client et serveur	lib	Le fichier ogspring.jar contient les classes de support de l'intégration de l'infrastructure SpringSource Spring.
wsogclient.jar	Local et client	lib	Fichier wsogclient.jar installé si vous utilisez un environnement qui contient WebSphere Application Server Version 6.0.2 et versions ultérieures. Ce fichier ne contient que les environnements d'exécution local et client.
wxssizeagent.jar	Local, client et serveur	lib	Le fichier wxssizeagent.jar permet de fournir des informations plus précises sur le dimensionnement des entrées de la mémoire cache lors de l'utilisation de l'environnement d'exécution Java (JRE) Version 1.5 ou ultérieure.
ibmcfw.jar ibmext.jar ibmorb.jar ibmorbapi.jar	Client et serveur	libendorsed	Cet ensemble de fichiers inclut l'environnement d'exécution de l'ORB (Object Request Broker) qui permet d'exécuter des applications dans des processus Java SE.
restservice.ear	Client	restservice/lib	Le fichier restservice.ear contient l'archive d'entreprise de l'application du service de données REST d'eXtreme Scale pour les environnements WebSphere Application Server.
restservice.war	Client	restservice/lib	Le fichier restservice.war contient l'archive Web du service de données REST d'eXtreme Scale pour les serveurs d'applications acquis auprès d'un autre fournisseur.
xsadmin.jar	Utilitaire	samples	Le fichier xsadmin.jar contient l'exemple d'utilitaire d'administration d'eXtreme Scale.
sessionobjectgrid.jar	Client et serveur	session/lib	Le fichier sessionobjectgrid.jar contient l'environnement d'exécution de gestion des sessions HTTP d'eXtreme Scale.
splicerlistener.jar	Utilitaire	session/lib	Le fichier splicerlistener.jar contient l'utilitaire splicer du filtre du gestionnaire des sessions HTTP d'eXtreme Scale.
splicer.jar	Utilitaire		

Procédure

1. Utilisez l'assistant pour effectuer l'installation.
   - Exécutez le script suivant pour démarrer l'installation intégrale de WebSphere eXtreme Scale :
     - Linux UNIX *racine_dvd*/install
     - Windows *racine_dvd*\install.bat
   - Exécutez le script suivant pour démarrer l'installation de WebSphere eXtreme Scale Client :
     - Linux UNIX root/*client_WXS*/install
     - Windows root/*client_WXS*\install.bat
2. Suivez les invites de l'assistant et cliquez sur **Terminer**.

   Restriction : Le panneau des fonctions dispositives répertorie les fonctions que vous pouvez choisir d'installer. Toutefois, des fonctions ne peuvent pas être ajoutées de manière incrémentielle à l'environnement du produit une fois que le produit a été installé. Si vous choisissez de ne pas installer une fonction lors de l'installation initiale du produit, vous devez désinstaller, puis réinstaller le produit pour l'ajouter.

Que faire ensuite

Pour configurer vos processus d'application client et vos processus serveur, reportez-vous à la configuration d'eXtreme Scale.

Intégration de WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client à WebSphere Application Server

Vous pouvez installer WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client dans un environnement dans lequel WebSphere Application Server ou WebSphere Application Server Network Deployment est installé. Vous pouvez utiliser les fonctions existantes de WebSphere Application Server ou WebSphere Application Server Network Deployment pour améliorer vos applications eXtreme Scale.

Avant de commencer

- Installez WebSphere Application Server ou WebSphere Application Server Network Deployment. Pour plus d'informations, voir Installation de l'environnement de traitement des applications.
- En fonction de la version que vous installez (Version 6.0.x, Version 6.1 ou Version 7.0), appliquez le dernier groupe de correctifs de WebSphere Application Server ou WebSphere Application Server Network Deployment pour mettre à jour le niveau de votre produit. Pour plus d'informations, reportez-vous aux groupes de correctifs les plus récents de WebSphere Application Server.
- Vérifiez que le répertoire d'installation cible ne contient pas une installation de WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client.
- Arrêtez tous les processus en cours d'exécution dans votre environnement WebSphere Application Server ou WebSphere Application Server Network Deployment. Pour en savoir plus sur les commmandes stopManager, stopNode et stopServer, voir Using wsadmin scripting command line tools.

ATTENTION :
Vérifiez que les processus en cours d'exécution sont arrêtés. Si les processus en cours d'exécution ne sont pas arrêtés, l'installation se poursuit et crée des résultats imprévisibles, ce qui la laisse dans un état indéterminé sur certaines plateformes.

Important : Lorsque vous installez WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client, il doit se trouver dans le répertoire dans lequel vous avez installé WebSphere Application Server. Par exemple, si vous avez installé WebSphere Application Server dans `C:\<racine_was>`, vous devez également choisir `C:<racine_was>` comme répertoire cible pour votre installation WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client.

Pourquoi et quand exécuter cette tâche

Intégrez eXtreme Scale à WebSphere Application Server ou WebSphere Application Server Network Deployment pour appliquer les fonctions d'eXtreme Scale à vos applications Java Platform, Enterprise Edition. Les applications Java EE hébergent les grilles de données et y accèdent à l'aide d'une connexion client.

Tableau 3. Fichiers d'exécution de WebSphere eXtreme Scale. Le tableau ci-après répertorie les fichiers JAR (archives Java) inclus dans l'installation.

Nom de fichier	Environnement	Emplacement d'installation	Description
`wxsdynacache.jar`	Client et serveur	`lib`	Le fichier `wxsdynacache.jar` contient les classes nécessaires à utiliser avec le fournisseur de mémoire cache dynamique.
`wsobjectgrid.jar`	Local et client	`lib`	Le fichier `wsobjectgrid.jar` contient les environnements d'exécution local, du serveur et du client eXtreme Scale.
`ogagent.jar`	Local, client et serveur	`lib`	Le fichier `ogagent.jar` contient les classes d'exécution requises pour exécuter l'agent d'instrumentation Java utilisé avec l'API EntityManager.
`ogsip.jar`	Client et serveur	`lib`	Le fichier `ogsip.jar` contient l'environnement d'exécution SIP (Session Initiation Protocol) d'eXtreme Scale.
`sessionobjectgrid.jar`	Client et serveur	`lib`	Le fichier `sessionobjectgrid.jar` contient l'environnement d'exécution de gestion des sessions HTTP d'eXtreme Scale.
`sessionobjectgridsip.jar`	Client et serveur	`lib`	Le fichier `sessionobjectgridsip.jar` contient l'environnement d'exécution de gestion des sessions SIP d'eXtreme Scale.
`wsogclient.jar`	Local et client	`lib`	Fichier `wsogclient.jar` installé si vous utilisez un environnement qui contient WebSphere Application Server Version 6.0.2 et versions ultérieures. Ce fichier ne contient que les environnements d'exécution local et client.
`wxssizeagent.jar`	Local, client et serveur	`lib`	Le fichier `wxssizeagent.jar` permet de fournir des informations plus précises sur le dimensionnement des entrées de la mémoire cache lors de l'utilisation de l'environnement d'exécution Java (JRE) Version 1.5 ou ultérieure.
`oghibernate-cache.jar`	Client et serveur	`optional Libraries/ ObjectGrid`	Le fichier `oghibernate-cache.jar` contient le plug-in de mémoire cache de niveau 2 d'eXtreme Scale pour JBoss Hibernate.
`ogspring.jar`	Local, client et serveur	`optional Libraries/ ObjectGrid`	Le fichier `ogspring.jar` contient les classes de support de l'intégration de l'infrastructure SpringSource Spring.
`xsadmin.jar`	Utilitaire	`optional Libraries/ ObjectGrid`	Le fichier `xsadmin.jar` contient l'exemple d'utilitaire d'administration d'eXtreme Scale.

`ibmcfw.jar`
`ibmext.jar`
`ibmorb.jar`
`ibmorbapi.jar`
	Client et serveur	`optional Libraries/ ObjectGrid/ endorsed`	Cet ensemble de fichiers inclut l'environnement d'exécution de l'ORB (Object Request Broker) qui permet d'exécuter des applications dans des processus Java SE.

Tableau 3. Fichiers d'exécution de WebSphere eXtreme Scale (suite). Le tableau ci-après répertorie les fichiers JAR (archives Java) inclus dans l'installation.

Nom de fichier	Environnement	Emplacement d'installation	Description
wxshyperic.jar	Utilitaire	optional Libraries/ ObjectGrid/ hyperic/ lib	Plug-in de détection du serveur WebSphere eXtreme Scale pour l'agent de surveillance SpringSource Hyperic.
restservice.ear	Client	optional Libraries/ ObjectGrid/ restservice/ lib	Le fichier restservice.ear contient l'archive d'entreprise de l'application du service de données REST d'eXtreme Scale pour les environnements WebSphere Application Server.
restservice.war	Client	optional Libraries/ ObjectGrid/ restservice/ lib	Le fichier restservice.war contient l'archive Web du service de données REST d'eXtreme Scale pour les serveurs d'applications acquis auprès d'un autre fournisseur.
splicerlistener.jar	Utilitaire	optional Libraries/ ObjectGrid/ session/ lib	Le fichier splicerlistener.jar contient l'utilitaire splicer du filtre du gestionnaire des sessions HTTP d'eXtreme Scale.
splicer.jar	Utilitaire	optional Libraries/ ObjectGrid/ legacy/ session/ lib	Le fichier splicer.jar contient l'utilitaire splicer Version 7.0 du filtre du gestionnaire des sessions HTTP d'eXtreme Scale.

Tableau 4. Fichiers d'exécution de WebSphere eXtreme Scale Client. Le tableau ci-après répertorie les fichiers JAR (archives Java) inclus dans l'installation.

Nom de fichier	Environnement	Emplacement d'installation	Description
wxsdynacache.jar	Client et serveur	lib	Le fichier wxsdynacache.jar contient les classes nécessaires à utiliser avec le fournisseur de mémoire cache dynamique.
ogagent.jar	Local, client et serveur	lib	Le fichier ogagent.jar contient les classes d'exécution requises pour exécuter l'agent d'instrumentation Java utilisé avec l'API EntityManager.
ogsip.jar	Client et serveur	lib	Le fichier ogsip.jar contient l'environnement d'exécution SIP (Session Initiation Protocol) d'eXtreme Scale.
sessionobjectgrid.jar	Client et serveur	lib	Le fichier sessionobjectgrid.jar contient l'environnement d'exécution de gestion des sessions HTTP d'eXtreme Scale.
sessionobjectgridsip.jar	Client et serveur	lib	Le fichier sessionobjectgridsip.jar contient l'environnement d'exécution de gestion des sessions SIP d'eXtreme Scale.
wsogclient.jar	Local et client	lib	Fichier wsogclient.jar installé si vous utilisez un environnement qui contient WebSphere Application Server Version 6.0.2 et versions ultérieures. Ce fichier ne contient que les environnements d'exécution local et client.
wxssizeagent.jar	Local, client et serveur	lib	Le fichier wxssizeagent.jar permet de fournir des informations plus précises sur le dimensionnement des entrées de la mémoire cache lors de l'utilisation de l'environnement d'exécution Java (JRE) Version 1.5 ou ultérieure.
oghibernate-cache.jar	Client et serveur	optional Libraries/ ObjectGrid	Le fichier oghibernate-cache.jar contient le plug-in de mémoire cache de niveau 2 d'eXtreme Scale pour JBoss Hibernate.
ogspring.jar	Local, client et serveur	optional Libraries/ ObjectGrid	Le fichier ogspring.jar contient les classes de support de l'intégration de l'infrastructure SpringSource Spring.
xsadmin.jar	Utilitaire	optional Libraries/ ObjectGrid	Le fichier xsadmin.jar contient l'exemple d'utilitaire d'administration d'eXtreme Scale.
ibmcfw.jar ibmext.jar ibmorb.jar ibmorbapi.jar	Client et serveur	optional Libraries/ ObjectGrid/ endorsed	Cet ensemble de fichiers inclut l'environnement d'exécution de l'ORB (Object Request Broker) qui permet d'exécuter des applications dans des processus Java SE.

Tableau 4. Fichiers d'exécution de WebSphere eXtreme Scale Client (suite). Le tableau ci-après répertorie les fichiers JAR (archives Java) inclus dans l'installation.

Nom de fichier	Environnement	Emplacement d'installation	Description
wxshyperic.jar	Utilitaire	optional Libraries/ ObjectGrid/ hyperic/ lib	Plug-in de détection du serveur WebSphere eXtreme Scale pour l'agent de surveillance SpringSource Hyperic.
restservice.ear	Client	optional Libraries/ ObjectGrid/ restservice/ lib	Le fichier restservice.ear contient l'archive d'entreprise de l'application du service de données REST d'eXtreme Scale pour les environnements WebSphere Application Server.
restservice.war	Client	optional Libraries/ ObjectGrid/ restservice/ lib	Le fichier restservice.war contient l'archive Web du service de données REST d'eXtreme Scale pour les serveurs d'applications acquis auprès d'un autre fournisseur.
splicerlistener.jar	Utilitaire	optional Libraries/ ObjectGrid/ session/ lib	Le fichier splicerlistener.jar contient l'utilitaire splicer du filtre du gestionnaire des sessions HTTP d'eXtreme Scale.
splicer.jar	Utilitaire	optional Libraries/ ObjectGrid/ legacy/ session/ lib	Le fichier splicer.jar contient l'utilitaire splicer Version 7.0 du filtre du gestionnaire des sessions HTTP d'eXtreme Scale.

Procédure

1. Utilisez l'assistant pour effectuer l'installation.
   - Exécutez le script suivant pour démarrer l'installation intégrale de WebSphere eXtreme Scale :
     - Linux UNIX *racine_dvd*/install
     - Windows *racine_dvd*\install.bat
   - Exécutez le script suivant pour démarrer l'installation de WebSphere eXtreme Scale Client :
     - Linux UNIX root/*client_WXS*/install
     - Windows root/*client_WXS*\install.bat
2. Suivez les invites de l'assistant.

   Le panneau des fonctions facultatives répertorie les fonctions que vous pouvez choisir d'installer. Toutefois, des fonctions ne peuvent pas être ajoutées de manière incrémentielle à l'environnement du produit une fois que le produit a été installé. Si vous choisissez de ne pas installer une fonction lors de l'installation initiale du produit, vous devez désinstaller, puis réinstaller le produit pour l'ajouter.

   Le panneau Extension de profil répertorie les profils existants que vous pouvez sélectionner pour étendre les fonctions d'eXtreme Scale. Si vous sélectionnez des profils existants déjà en cours d'utilisation, toutefois, un panneau d'avertissement est affiché. Pour poursuivre l'installation, arrêtez les serveurs configurés dans les profils ou cliquez sur **Précédent** pour supprimer les profils de votre sélection.

Que faire ensuite

Si vous exécutez WebSphere Application Server Version 6.1 ou 7.0, vous pouvez utiliser le plug-in de l'outil de gestion de profil ou la commande manageprofiles. Si vous exécutez WebSphere Application Server Version 6.0.2, vous devez utiliser la commande wasprofile pour créer et étendre des profils.

Déployez votre application, démarrez un service de catalogue et démarrez les conteneurs dans votre environnement WebSphere Application Server. Pour plus d'informations, voir «Administration de WebSphere eXtreme Scale avec WebSphere Application Server», à la page 344.

Utilisation du plug-in Installation Factory pour créer et installer des modules personnalisés

Utilisez le plug-in IBM Installation Factory pour que WebSphere eXtreme Scale crée un module d'installation personnalisée (CIP) ou un module d'installation intégrée (IIP). Un module d'installation personnalisée contient un unique module d'installation du produit et diverses ressources facultatives. Un module d'installation intégrée associe un ou plusieurs modules d'installation dans un même flux de travaux d'installation que vous concevez.

Avant de commencer

Avant de créer et d'installer des modules personnalisés pour eXtreme Scale, vous devez télécharger les produits suivants :

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

Pourquoi et quand exécuter cette tâche

A l'aide d'Installation Factory, vous pouvez créer un module CIP en associant un composant de produit à des modules de maintenance, des scripts de personnalisation et d'autres fichiers. Lorsque vous créez un module IIP, vous regrouper les composants individuels ou les modules d'installation dans un même module d'installation.

Fichier de définition de génération

Un fichier de définition de génération est un document XML qui spécifie comment créer et installer un module d'installation personnalisée (CIP) ou un module d'installation intégrée (IIP). IBM® Installation Factory for WebSphere eXtreme Scale lit les détails relatifs au module dans le fichier de définition de génération pour générer un module CIP ou IIP.

Pour pouvoir créer un module CIP ou IIP, vous devez créer un fichier de définition de génération pour chaque module personnalisé. Le fichier de définition de génération décrit les composants du produit ou les modules d'installation à installer, l'emplacement du module CIP ou IIP, les modules de maintenance à inclure, les scripts installation et les autres fichiers à intégrer. Vous pouvez également spécifier dans le fichier de définition de génération du module IIP l'ordre suivant lequel Installation Factory installe chaque module d'installation.

L'assistant de définition de génération vous guide à travers les étapes de création d'un fichier de définition de génération. Vous pouvez aussi utiliser l'assistant pour modifier un fichier de définition de génération existant. Chaque page de l'assistant vous invite à entrer des informations relatives à un module personnalisé telles que l'identification du module, l'emplacement d'installation de la définition de génération et l'emplacement d'installation du module personnalisé. Toutes ces informations sont enregistrées dans le nouveau fichier de définition de génération ou modifiées et enregistrées dans un fichier de définition de génération existant. Pour plus d'informations, consultez les sections Pages de l'assistant de définition de génération du module CIP et Pages de l'assistant de définition de génération du module IIP.

Pour créer le fichier de définition de génération uniquement, vous pouvez utiliser l'outil de l'interface de ligne de commande pour générer le module personnalisé en dehors de l'interface graphique. Pour plus d'informations, voir «Installation en mode silencieux d'un module CIP ou IIP», à la page 33.

Création d'un fichier de définition de génération et génération d'un module d'installation personnalisée (CIP)

Le plug-in IBM Installation Factory de WebSphere eXtreme Scale génère un module d'installation personnalisée (CIP) en fonction des détails que vous spécifiez dans le fichier de définition de génération. La définition de génération spécifie le module de produit à installer, l'emplacement du module CIP, les modules de maintenance à inclure dans l'installation, les fichiers du script d'installation et les fichiers supplémentaires à inclure dans le module CIP.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser l'Assistant de définition des générations pour créer un fichier de définition de génération et générer un module CIP.

Procédure

1. Exécutez le script suivant à partir du répertoire *REP_BASE_IF*/bin pour démarrer Installation Factory :
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`

   Cliquez sur l'icône **Nouvelle définition de génération**.
2. Sélectionnez le produit à inclure dans le fichier de définition de génération et cliquez sur **Terminer** pour démarrer l'assistant de définition de génération.
3. Suivez les invites de l'assistant.

   Dans le panneau Scripts d'installation et de désinstallation, cliquez sur **Ajout des scripts...** pour alimenter la table avec des scripts d'installation personnalisés. Saisissez l'emplacement des fichiers script et désélectionnez la case à cocher permettant de continuer si un message d'erreur s'affiche. L'opération est arrêtée par défaut. Cliquez sur **OK** pour retourner au panneau.

Résultats

Vous avez créé et personnalisé le fichier de définition de génération et généré le module CIP si vous avez choisi de travailler en mode connecté.

Si l'assistant de définition de génération ne vous permet pas de générer le module CIP à partir du fichier de définition de génération, vous pouvez toujours le générer en exécutant le script `ifcli.sh|bat` à partir du répertoire *REP_BASE_IF*/bin.

Que faire ensuite

Installez le module CIP. Pour plus d'informations, voir «Installation d'un module CIP».

Installation d'un module CIP :

Simplifiez la procédure d'installation du produit en installant un module d'installation personnalisée (CIP). Un module CIP est une image d'installation de produit unique qui peut inclure un ou plusieurs modules de maintenance, des scripts de configuration et d'autres fichiers.

Avant de commencer

Avant d'installer un module CIP, vous devez créer un fichier de définition de génération pour spécifier les options à inclure dans le module CIP. Pour plus d'informations, voir «Création d'un fichier de définition de génération et génération d'un module d'installation personnalisée (CIP)», à la page 27.

Pourquoi et quand exécuter cette tâche

Un module CIP associe un composant de produit à des modules de maintenance, des scripts de personnalisation et d'autres fichiers, puis l'installe.

Procédure

1. Arrêtez tous les processus en cours d'exécution sur le poste de travail que vous préparez pour l'installation. Pour arrêter le gestionnaire de déploiement, exécutez le script suivant :
   - Linux UNIX *racine_profil*/bin/stopManager.sh
   - Windows *racine_profil*\bin\stopManager.bat

   Pour arrêter les noeuds, exécutez le script suivant :
   - Linux UNIX *racine_profil*/bin/stopNode.sh
   - Windows *racine_profil*\bin\stopNode.bat
2. Exécutez le script suivant pour démarrer l'installation :
   - Linux UNIX *rép_base_CIP*/bin/install
   - Windows *rép_base_CIP*\bin\install.bat
3. Suivez les invites de l'assistant pour effectuer l'installation.

   Le panneau des fonctions facultatives répertorie les fonctions que vous pouvez choisir d'installer. Toutefois, des fonctions ne peuvent pas être ajoutées de manière incrémentielle à l'environnement du produit une fois que le produit a été installé. Si vous choisissez de ne pas installer une fonction lors de l'installation initiale du produit, vous devez désinstaller, puis réinstaller le produit pour l'ajouter.

   Le panneau Extension de profil répertorie les profils existants que vous pouvez sélectionner pour étendre les fonctions d'eXtreme Scale. Si vous sélectionnez des profils existants déjà en cours d'utilisation, toutefois, un panneau d'avertissement est affiché. Pour poursuivre l'installation, arrêtez les serveurs configurés dans les profils ou cliquez sur **Précédent** pour supprimer les profils de votre sélection.

Résultats

Vous avez installé le module CIP.

Que faire ensuite

Si vous exécutez WebSphere Application Server Version 6.1 ou 7.0, vous pouvez utiliser le plug-in de l'outil de gestion de profil ou la commande manageprofiles pour créer et étendre des profils. Si vous exécutez WebSphere Application Server Version 6.0.2, vous devez utiliser la commande wasprofile pour créer et étendre des profils. Pour plus d'informations, voir «Création et augmentation de profils pour WebSphere eXtreme Scale», à la page 42.

Si vous avez étendu les profils de eXtreme Scale lors de la procédure d'installation, vous pouvez déployer des applications, démarrer un service de catalogue et démarrer les conteneurs de votre environnement WebSphere Application Server. Pour plus d'informations, voir «Administration de WebSphere eXtreme Scale avec WebSphere Application Server», à la page 344.

Installation d'un module CIP pour appliquer la maintenance à une installation de produit existante :

Vous pouvez appliquer des modules de maintenance à une installation de produit existante en installant un module d'installation personnalisée (CIP). La procédure d'application de la maintenance à une installation existante avec un module est couramment appelée *installation intermédiaire*.

Avant de commencer

Créez un fichier de définition de génération pour spécifier les options à inclure dans le module CIP. Pour plus d'informations, voir «Création d'un fichier de définition de génération et génération d'un module d'installation personnalisée (CIP)», à la page 27.

Pourquoi et quand exécuter cette tâche

Lorsque vous appliquez la maintenance avec un module CIP qui contient un groupe de mises à jour, un groupe de correctifs ou les deux, tous les correctifs APAR précédemment installés sont désinstallés par l'assistant. Si le module CIP est au même niveau que le produit, les correctifs APAR installés précédemment ne sont conservés que s'ils sont intégrés au module CIP. Pour appliquer correctement la maintenance à une installation existante, vous devez inclure les fonctions installées dans le module CIP.

Procédure

1. Arrêtez tous les processus en cours d'exécution sur le poste de travail que vous préparez pour l'installation. Pour arrêter le gestionnaire de déploiement, exécutez le script suivant :
   - Linux UNIX *racine_profil*/bin/stopManager.sh
   - Windows *racine_profil*\bin\stopManager.bat

   Pour arrêter les noeuds, exécutez le script suivant :
   - Linux UNIX *racine_profil*\bin\stopNode.sh
   - Windows *racine_profil*\bin\stopNode.bat
2. Exécutez le script suivant pour démarrer l'installation :
   - Linux UNIX *rép_base_CIP*/bin/install
   - Windows *rép_base_CIP*\bin\install.bat
3. Suivez les invites de l'assistant pour effectuer l'installation.

   Le récapitulatif de l'aperçu d'installation répertorie la version de produit résultante et les éventuels fonctions et correctifs temporaires applicables. Ensuite, l'assistant applique la maintenance et met à jour les fonctions du produit.

Résultats

Les fichiers binaires du produit sont copiés dans le répertoire *rép_base_was*/properties/version/nif/backup. Vous pouvez utiliser IBM Update Installer pour désinstaller la mise à jour et restaurer votre poste de travail. Pour plus d'informations, voir «Désinstallation de mises à jour de module CIP d'une installation de produit existante.».

Désinstallation de mises à jour de module CIP d'une installation de produit existante. :

Vous pouvez supprimer des mises à jour de module CIP d'une installation de produit existante sans supprimer l'intégralité du produit. Utilisez IBM Update Installer Version 7.0.0.4 pour désinstaller les mises à jour de module CIP. Cette tâche est également appelée *désinstallation intermédiaire*.

Avant de commencer

Au moins une copie du produit doit être installé sur le système.

Procédure

1. Téléchargez la version 7.0.0.4 du programme d'installation de mises à jour à partir du site FTP suivant :

   ftp://ftp.software.ibm.com/software/websphere/cw/process_server/FEP/UPDI/7004
2. Installez le programme d'installation de mises à jour. Pour plus d'informations, voir la rubrique Installation d'Update Installer pour WebSphere Software dans le Centre de documentation de WebSphere Application Server.
3. Désinstallez les éventuels groupes de correctifs, groupes de mises à jour ou correctifs temporaires que vous avez ajoutés à votre environnement après avoir installé le module CIP.
4. Désinstallez les éventuels correctifs que vous avez inclus dans l'installation intermédiaire. Cette procédure est identique à la désinstallation d'un groupe de correctifs ou d'un groupe de mises à jour. Toutefois, la maintenance incluse dans le module CIP est maintenant incluse dans une opération unique.
5. Désinstallez le module CIP à l'aide du programme d'installation de mises à jour. Les niveaux de maintenance retournent à leur état avant mise à jour et le module CIP est dénoté par l'identificateur CIP qui est ajouté comme préfixe à son nom de fichier. L'exemple suivant montre comment un module CIP est affiché de manière différente des autres modules de maintenance standard sur le panneau de sélection des modules de maintenance :

   CIP

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

Résultats

Vous avez supprimé les mises à jour de module CIP d'une installation de produit existante.

Création d'un fichier de définition de génération et génération d'un module IIP

Le plug-in IBM Installation Factory de WebSphere eXtreme Scale génère un module IIP en fonction des propriétés fournies par le fichier de définition de génération. Le fichier de définition de génération contient des informations telles que les modules

d'installation à inclure dans le module IIP, l'ordre suivant lequel Installation Factory installe chaque module et l'emplacement du module IIP.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser l'Assistant de définition des générations pour créer un fichier de définition de génération et générer un module IIP.

Procédure

1. Exécutez le script suivant à partir du répertoire *REP_BASE_IF*/bin pour démarrer Installation Factory :
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`
2. Cliquez sur l'icône **Create New Integrated Installation Package** pour démarrer l'assistant de définition de génération.
3. Suivez les invites de l'assistant.
   a. Dans le panneau Construct the IIP, sélectionnez un pris en charge pris en charge dans la liste, puis cliquez sur **Add Installer** pour ajouter le module d'installation au module IIP. Un panneau indiquant le nom du module, son identificateur et ses propriétés s'affiche. Pour afficher des informations spécifiques sur le module sélectionné, cliquez sur **View Installation Package Information**. Cliquez sur **Modifier** pour entrer le chemin de répertoire du module d'installation pour chaque système d'exploitation. Si vous ajoutez actuellement un module d'installation pour WebSphere Extended Deployment, cochez la case qui vous permet d'utiliser le même module pour tous les systèmes d'exploitation pris en charge. Cliquez sur **OK** et retournez au panneau Construct the IIP. Un appel est créé par défaut.
      - Pour modifier le chemin de répertoire d'un module d'installation, sélectionnez le module dans la liste des modules d'installation utilisés dans le module IIP, puis cliquez sur **Modifier**.
      - Pour modifier un appel, sélectionnez-le et cliquez sur **Modifier**. Spécifiez l'emplacement d'installation par défaut de l'appel sur chaque système d'exploitation. Spécifiez l'emplacement du fichier de réponses si vous sélectionnez une installation en mode silencieux comme mode d'installation par défaut.
      - Cliquez sur **Add Invocation** pour ajouter une contribution d'appel au module d'installation. Un panneau à partir duquel vous pouvez spécifier les propriétés de l'appel s'affiche.
      - Cliquez sur **Supprimer** pour supprimer des modules d'installation ou des appels.
4. Vérifiez le récapitulatif de vos sélections, sélectionnez l'option **Enregistrer le fichier de définition de génération et générer le module d'installation personnalisé** et cliquez sur **Terminer**.

   Vous pouvez également sauvegarder le fichier de définition de génération sans générer le module IIP. Avec cette option, vous générez en fait le module IIP en dehors de l'assistant en exécutant le script `ifcli.bat | ifcli.sh` à partir du répertoire *rép_base_IF*/bin/.

Résultats

Vous avez créé et personnalisé le fichier de définition de génération d'un module IIP.

Que faire ensuite

Installez le module IIP.

Installation d'un module IIP :

Utilisez le plug-in IBM Installation Factory pour que WebSphere eXtreme Scale installe un module d'installation intégrée (IIP). Un module d'installation intégrée associe un ou plusieurs modules d'installation dans un même flux de travaux que vous concevez.

Avant de commencer

Avant d'installer un module CIP, vous devez créer un fichier de définition de génération pour spécifier les options à inclure dans le module CIP. Pour plus d'informations, voir «Création d'un fichier de définition de génération et génération d'un module IIP», à la page 30.

Pourquoi et quand exécuter cette tâche

Un module IIP peut inclure un ou plusieurs modules d'installation à disponibilité générale, un ou plusieurs modules CIP et d'autres fichiers et répertoires facultatifs. En installant un module IIP, vous regroupez plusieurs modules d'installation, ou *contributions*, dans un même module, puis vous installez ces contributions suivant un ordre spécifique pour effectuer une installation de bout en bout.

Procédure

1. Exécutez le script suivant pour démarrer l'assistant :
   - Linux UNIX *rép_base_IIP*/bin/install
   - Windows *rép_base_IIP*\bin\install.bat
2. Cliquez sur **A propos de** dans le panneau de bienvenue pour afficher les détails du module IIP, tels que l'identificateur du module, les systèmes d'exploitation pris en charge et les modules d'installation inclus.

   Facultatif : Pour modifier les options d'installation de chaque package, cliquez sur **Modifier**.

   Facultatif : Deux boutons **Afficher le journal** sont affichés dans le panneau de l'assistant. Pour afficher le journal de chaque module, cliquez sur le bouton **Afficher le journal** affiché en regard du tableau qui répertorie les modules d'installation. Pour afficher les informations générales du journal du module IIP, cliquez sur le bouton **Afficher le journal** affiché en regard des informations de statut.
3. Sélectionnez les modules d'installation à exécuter, puis cliquez sur **Installer**. Une liste de toutes les contributions suivant leur ordre d'appel que le module IIP contient est affichée. Pour indiquer quels appels de contribution ne doivent pas être exécutés lors de l'installation, désélectionnez la case à cocher en regard de la zone **Nom de l'installation**.

Résultats

Vous avez installé un module IIP.

Modification d'un fichier de définition de génération existant pour un module IIP :

Vous pouvez éditer les propriétés d'un module IIP ou en ajouter pour personnaliser davantage l'installation.

Pourquoi et quand exécuter cette tâche

Pour modifier les propriétés d'un module IIP, modifiez le fichier de définition de génération existant.

Procédure

1. Exécutez le script suivant à partir du répertoire *REP_BASE_IF*/bin pour démarrer Installation Factory :
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`
2. Cliquez sur l'icône d'**ouverture de la définition de génération** et sélectionnez le fichier de définition de génération à modifier.
3. Sélectionnez les propriétés spécifiques du module IIP à modifier. La liste suivante répertorie les modifications possibles que vous pouvez effectuer :
   - Modifiez votre sélection de mode actuelle. En mode connecté, vous créez la définition de génération à utiliser et générez éventuellement le module IIP, à partir de votre poste de travail actuel. En mode déconnecté, vous créez le fichier de définition de génération à utiliser sur un autre poste de travail.
   - Ajoutez ou supprimez les systèmes d'exploitation existants que le module IIP prend en charge.
   - Editez l'identificateur et la version existants du module IIP.
   - Editez l'emplacement cible du fichier de définition de génération.
   - Editez l'emplacement cible du module IIP.
   - Choisissez d'afficher ou non un assistant d'installation pour le module IIP. L'assistant fournit des informations sur le module IIP et les options d'installation lorsque le module IIP est exécuté.
   - Ajoutez, supprimez et éditez les modules d'installation qui se trouvent dans le module IIP.

     Important : Si vous avez ajouté un système d'exploitation pris en charge et que vous n'avez pas mis à jour les propriétés du module d'installation dans le module IIP, vous recevez un message d'avertissement indiquant que les contributions sélectionnées ne contiennent pas de modules d'installation identifiés pour tous les systèmes d'exploitation pris en charge par le module IIP. Cliquez sur **Oui** pour continuer ou sur **Non** pour éditer le module d'installation.
4. Vérifiez le récapitulatif de vos sélections, sélectionnez **Enregistrer le fichier de définition de génération et générer le module d'installation personnalisé**, puis cliquez sur **Terminer**.

Installation en mode silencieux d'un module CIP ou IIP

Vous pouvez installer en mode silencieux un module d'installation personnalisée (CIP) ou un module d'installation intégrée (IIP) pour le produit à l'aide d'un fichier de réponses complet, que vous configurez spécifiquement en fonction de vos besoins, ou de paramètres que vous transmettez à la ligne de commande.

Avant de commencer

Créez le fichier de définition de génération du module CIP ou IIP. Pour plus d'informations, voir «Création d'un fichier de définition de génération et génération d'un module d'installation personnalisée (CIP)», à la page 27.

Pourquoi et quand exécuter cette tâche

Une installation en mode silencieux utilise le même programme d'installation que l'interface graphique. Toutefois, au lieu d'afficher une interface d'assistant, elle lit toutes vos réponses dans un fichier que vous personnalisez ou dans les paramètres que vous transmettez à la ligne de commande. Si vous installez en mode silencieux un module IIP, vous pouvez appeler une contribution avec une combinaison d'options que vous spécifiez directement sur la ligne de commande, ainsi que des options que vous spécifiez dans un fichier de réponses. Toutefois, si vous transmettez des options de contribution à la ligne de commande, le programme d'installation du module IIP ignore toutes les options spécifiées dans le fichier de réponses d'une contribution spécifique. Pour plus d'informations, reportez-vous à la rubrique Installation d'un package d'installation intégrée en mode silencieux.

Remarque : Vous devez spécifier le nom complet du fichier de réponses. Si vous spécifiez le chemin relatif, l'installation échoue sans message d'erreur.

Procédure

1. Facultatif : Si vous choisissez d'installer le module CIP ou IIP à l'aide d'un fichier de réponses, commencez par personnaliser ce fichier.
   a. Copiez le fichier de réponses, `wxssetup.response.txt`, du DVD du produit sur votre unité de disque.
   b. Ouvrez et éditez le fichier de réponses dans l'éditeur de texte de votre choix. Le fichier inclut des commentaires pour faciliter la procédure de configuration et doit inclure ces paramètres :
      - Contrat de licence
      - Emplacement de l'installation du produit

      Conseil : Le programme d'installation utilise l'emplacement que vous sélectionnez pour votre installation pour déterminer où votre instance WebSphere Application Server est installée. Si vous effectuez l'installation sur un noeud qui contient plusieurs instances WebSphere Application Server, définissez précisément votre emplacement.
   c. Exécutez le script suivant pour démarrer votre fichier de réponses personnalisé.
      - Linux UNIX `install -options /chemin_absolu/fichier_réponses.txt -silent`
      - Windows `install.bat -options C:\chemin_unité\fichier_réponses.txt -silent`
2. Facultatif : Si vous choisissez d'installer le module CIP ou IIP en transmettant certains paramètres à la ligne de commande, exécutez le script suivant pour démarrer l'installation :
   - Linux UNIX `install -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=emplacement_install`

- Windows `install.bat -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=`*`emplacement_install`*

  où *emplacement_install* représente l'emplacement de votre installation WebSphere Application Server existante.

3. Recherchez les éventuelles erreurs ou un incident d'installation dans les journaux résultats.

Résultats

Vous avez installé le module CIP ou IIP.

Que faire ensuite

Si vous exécutez WebSphere Application Server Version 6.1 ou 7.0, vous pouvez utiliser le plug-in de l'outil de gestion de profil ou la commande manageprofiles pour créer et étendre des profils. Si vous exécutez WebSphere Application Server Version 6.0.2, vous devez utiliser la commande wasprofile pour créer et étendre des profils.

Si vous avez étendu les profils de eXtreme Scale lors de la procédure d'installation, vous pouvez déployer des applications, démarrer un service de catalogue et démarrer les conteneurs de votre environnement WebSphere Application Server. Pour plus d'informations, voir «Administration de WebSphere eXtreme Scale avec WebSphere Application Server», à la page 344.

Fichier `wxssetup.response.txt` :

Vous pouvez utiliser un fichier de réponses complet pour installer WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client en mode silencieux.

Fichier de réponses d'une installation intégrale de WebSphere eXtreme Scale

```
################################################################################
#
# Fichier d'options InstallShield d'IBM WebSphere eXtreme Scale V7.1.0
#
# Nom de l'assistant : Install
# Source de l'assistant : setup.jar
#
# Ce fichier peut être utilisé pour configurer Install avec les options
# spécifiées ci-après lorsque l'assistant est exécuté avec l'option de
# ligne de commande "-options". Consultez la documentation de chacun des
# paramètres pour obtenir des informations sur la manière de modifier sa
# valeur.
# Placez toutes les valeurs entre guillemets.
#
# Une utilisation courante d'un fichier d'options consiste à exécuter
# l'assistant en mode silencieux. Le créateur du fichier d'options peut ainsi
# spécifier les paramètres de l'assistant sans avoir à exécuter l'assistant
# en mode graphique ou console. Pour utiliser ce fichier d'options en vue d'une
# exécution en mode silencieux, utilisez les arguments de ligne de commande
# suivants lorsque vous exécutez l'assistant :
#
#    -options "D:\installImage\WXS\wxssetup.response" -silent
#
# Notez que le nom de fichier de réponses complet doit être utilisé.
#
################################################################################

################################################################################
#
```

```
# Acceptation de la licence
#
# Valeurs admises :
#  true - Accepte la licence. Installe le produit.
#  false - Refuse la licence. L'installation n'est pas effectuée.
#
# Si aucune installation n'est effectuée, cela est consigné dans un fichier journal
# temporaire, dans le répertoire temporaire de l'utilisateur.
#
# En spécifiant la valeur "true" pour la propriété silentInstallLicenseAcceptance
# dans ce fichier de réponses, vous certifiez que vous avez vérifié et que vous
# acceptez les dispositions des Conditions Internationales d'Utilisation des
# Logiciels IBM accompagnant ce programme, qui se trouve dans le répertoire
# RACINE_CD\XD\wxs.primary.pak\repository\legal.xs\license.xs.  Si vous les refusez,
# ne modifiez pas la valeur de cette propriété, ne téléchargez pas ce
# programme, ne l'installez pas, ne le copiez pas et n'y accédez pas. Retournez
# rapidement le programme et votre autorisation d'utilisation du logiciel au
# tiers auprès duquel vous l'avez acquis afin d'en obtenir le remboursement.
#
-OPT silentInstallLicenseAcceptance="false"


###############################################################################
# Vérification des prérequis non bloquants
#
# Si vous souhaitez désactiver la vérification des prérequis non bloquants,
# supprimez la mise en commentaire de la ligne ci-après. Cette action indique
# au programme d'installation de poursuivre l'installation et de consigner les
# avertissements bien que la vérification des prérequis ait échoué.
#
#-OPT disableNonBlockingPrereqChecking="true"


###############################################################################
#
# Emplacement d'installation
#
# Emplacement d'installation du produit. Spécifiez un répertoire valide dans
# lequel le produit doit être installé. Si le répertoire contient des espaces,
# placez-le entre guillemets, comme indiqué dans l'exemple Windows ci-après. Notez
# que les espaces dans l'emplacement d'installation ne sont pris en charge que
# sur les systèmes d'exploitation Windows. La longueur maximale du chemin d'accès
# est de 60 caractères pour Windows.
#
# Vous trouverez ci-après une liste des emplacements d'installation par défaut
# pour chaque système d'exploitation pris en charge si vous effectuez
# l'installation en tant qu'utilisateur root.  Par défaut, dans ce fichier de
# réponses, l'emplacement d'installation de Windows est utilisé.  Si vous
# souhaitez utiliser l'emplacement d'installation par défaut d'un autre
# système d'exploitation, supprimez la mise en commentaire de l'entrée
# d'emplacement d'installation par défaut (en supprimant le signe '#'), puis
# placez en commentaire (en ajoutant le signe '#') l'entrée de système
# d'exploitation Windows ci-après.
#
# L'emplacement d'installation est utilisé pour déterminer si WebSphere eXtreme
# Scale doit être installé comme déploiement autonome ou s'il doit être intégré
# à une installation WebSphere Application Server existante.
#
# Si l'emplacement spécifié correspond à une installation WebSphere Application
# Server ou WebSphere Network Deployment existante, eXtreme Scale est intégré
# au serveur WebSphere Application Server existant.  Si l'emplacement spécifié
# correspond à un répertoire nouveau ou vide, WebSphere eXtreme Scale est
# installé comme déploiement autonome.
#
# Remarque : Si l'emplacement d'installation spécifié contient une installation de
# WebSphere eXtreme Scale, WebSphere eXtended Deployment DataGrid ou
# ObjectGrid, l'installation échoue.
```

```
#
# Emplacement d'installation par défaut sous AIX :
#
#  -OPT installLocation="/usr/IBM/WebSphere/eXtremeScale"
#
# Emplacement d'installation par défaut sous HP-UX, Solaris ou Linux :
#
#  -OPT installLocation="/opt/IBM/WebSphere/eXtremeScale"
#
#
# Emplacement d'installation par défaut sous Windows :
#
-OPT installLocation="C:\Program Files\IBM\WebSphere\eXtremeScale"

#
# Si vous effectuez l'installation en tant qu'utilisateur autre que root sous
# Unix ou qu'administrateur sous Windows, les emplacements d'installation par
# défaut ci-après sont recommandés. Assurez-vous de disposer des droits d'accès
# en écriture pour l'emplacement d'installation choisi.
#
# Emplacement d'installation par défaut sous AIX :
#
#  -OPT installLocation="<répertoire de base de
# l'utilisateur>/IBM/WebSphere/eXtremeScale"

# Emplacement d'installation par défaut sous HP-UX, Solaris ou Linux :
#
#  -OPT installLocation="<répertoire de base de
# l'utilisateur>/IBM/WebSphere/eXtremeScale"
# Emplacement d'installation par défaut sous Windows :
#
# -OPT installLocation="C:\IBM\WebSphere\eXtremeScale"


################################################################################
#  Installation de fonctions facultatives
#
# Spécifiez les fonctions facultatives à installer en affectant à chacune
# d'elles la valeur "true".  Affectez aux fonctions facultatives que vous ne
# souhaitez pas installer la valeur "false".
#
# Les options selectServer, selectClient, selectPF et selectXSStreamQuery ne sont
# valides que si l'option installLocation ci-dessus contient une installation de
# WebSphere Application Server.  Ces options sont ignorées sur une
# installation autonome de WebSphere eXtreme Scale.
#
# Sur l'installation autonome de WebSphere eXtreme Scale, le serveur et le
# client eXtreme Scale sont automatiquement installés.  Les options de
# l'installation autonome d'eXtreme Scale sont selectXSConsoleOther et
# selectXSStreamQueryOther.

#
# Cette option, si elle est sélectionnée, installe les composants nécessaires
# pour exécuter des serveurs WebSphere eXtreme Scale et le fournisseur de service
# de mémoire cache dynamique eXtreme Scale. Si cette option est sélectionnée,
# le client WebSphere eXtreme Scale doit également être sélectionné
# en supprimant sa mise en commentaire et en lui affectant la valeur "true".
# Sinon, l'installation en mode silencieux ECHOUE.
#
-OPT selectServer="true"

#
# Cette option, si elle est sélectionnée, installe les composants nécessaires pour
# exécuter des applications client WebSphere eXtreme Scale. Si l'option Serveur est
# sélectionnée ci-dessus, cette option doit également être sélectionnée
# en supprimant sa mise en commentaire et en lui affectant la valeur "true". Sinon,
# l'installation en mode silencieux ECHOUE.
```

```
#
-OPT selectClient="true"

#
# Cette option, si elle est sélectionnée, installe les composants nécessaires pour
# exécuter la console WebSphere eXtreme Scale. Si cette option est sélectionnée,
# l'emplacement d'installation spécifié ci-dessus doit correspondre à un répertoire
# nouveau ou vide car l'option de la console n'est valide que pour un déploiement
# WebSphere eXtreme Scale autonome. Pour installer cette option,
# la mise en commentaire de la ligne d'option ci-après doit être supprimée et
# vous devez lui affecter la valeur "true".
#-OPT selectXSConsoleOther="false"

#
# Les options suivantes, si elles sont sélectionnées installeront la
# fonctionnalité OBSOLETE.
#
# Cette option sélectionne WebSphere Partition Facility pour l'installer.
# Cette fonctionnalité est OBSOLETE.  Pour installer cette option,
# la mise en commentaire de la ligne d'option ci-après doit être supprimée et
# vous devez lui affecter la valeur "true".
#
#-OPT selectPF="false"

#
# Cette option sélectionne WebSphere eXtreme Scale StreamQuery for WAS pour
# l'installation.  Cette fonctionnalité est OBSOLETE.  Pour installer cette option,
# la mise en commentaire de la ligne d'option ci-après doit être supprimée et
# vous devez lui affecter la valeur "true".
# Si cette option est sélectionnée, le client WebSphere
# eXtreme Scale doit également être sélectionné en supprimant sa mise en
# commentaire et en lui affectant la valeur "true".
# Sinon, l'installation en mode silencieux ECHOUE.
#
#-OPT selectXSStreamQuery="false"

#
# Cette option sélectionne WebSphere eXtreme Scale StreamQuery for J2SE pour
# l'installation.  Cette fonctionnalité est OBSOLETE.  Pour installer cette option,
# la mise en commentaire de la ligne d'option ci-après doit être supprimée et
# vous devez lui affecter la valeur "true".
# Si cette option est sélectionnée, le client WebSphere
# eXtreme Scale doit également être sélectionné en supprimant sa mise en
# commentaire et en lui affectant la valeur "true".
# Sinon, l'installation en mode silencieux ECHOUE.
#
#-OPT selectXSStreamQueryOther="false"


###############################################################################
#  Liste des profils à étendre
#
# Spécifiez les profils existants à étendre ou placez en commentaire la ligne
# permettant d'étendre tous les profils existants détectés par l'installation.
#
# Pour spécifier plusieurs profils, utilisez une virgule afin de séparer les
# différents noms de profil.
# Par exemple, "AppSrv01,Dmgr01,Custom01". La liste ne doit pas contenir d'espace.
#
-OPT profileAugmentList=""


###############################################################################
# Contrôle du traçage
#
# Le format de la sortie de trace peut être contrôlé via l'option
# -OPT traceFormat=ALL
```

```
#
# Les choix de format sont 'text' et 'XML'. Par défaut, ces deux formats seront
# générés, dans deux fichiers de trace différents.
#
# Si un seul format est requis, utilisez l'option traceFormat pour le spécifier,
# comme suit :
#
# Valeurs admises :
#
#  text -  Les lignes du fichier de trace sont au format texte pour une meilleure
#          lisibilité.
#  XML  -  Les lignes du fichier de trace se trouveront au format XML de
#          consignation Java standard qui peut être affiché à l'aide de tout
#          éditeur de texte ou éditeur XML ou à l'aide de l'outil Chainsaw
#          d'Apache, à l'adresse URL suivante :
#          (http://logging.apache.org/log4j/docs/chainsaw.html).
#
# La quantité d'informations de trace capturées peut être contrôlée à l'aide de
# l'option suivante :
# -OPT traceLevel=INFO
#
# Valeurs admises :
#
# Niveau    Niveau
# de trace  numérique  Description
# -------  ---------  ------------------------------------------------------
# OFF         0       Aucun fichier de trace n'est généré
# SEVERE      1       Seules les erreurs graves sont consignées dans le
#                      fichier de trace
# WARNING     2       Les messages relatifs aux exceptions non fatales et aux
#                     avertissements sont ajoutés au fichier de trace
# INFO        3       Les messages d'information sont ajoutés au fichier de trace
#                     (niveau de trace par défaut)
# CONFIG      4       Les messages liés à la configuration sont ajoutés
#                     au fichier de trace
# FINE        5       Traçage des appels de méthode pour les méthodes publiques
# FINER       6       Traçage des appels de méthode pour les méthodes non
#                     publiques, exceptées les méthodes d'accès get et set
# FINEST      7       Traçage de tous les appels de méthode ; l'entrée/la sortie
#                     de trace inclut les paramètres et la valeur de retour
```

Fichier de réponses d'une installation WebSphere eXtreme Scale Client

```
################################################################################
#
# Fichier d'options InstallShield d'IBM WebSphere eXtreme Scale Client V7.1.0
#
# Nom de l'assistant : Install
# Source de l'assistant : setup.jar
#
# Ce fichier peut être utilisé pour configurer Install avec les options
# spécifiées ci-après lorsque l'assistant est exécuté avec l'option de
# ligne de commande "-options". Consultez la documentation de chacun des
# paramètres pour obtenir des informations sur la manière de modifier sa
# valeur.
# Placez toutes les valeurs entre guillemets.
#
# Une utilisation courante d'un fichier d'options consiste à exécuter
# l'assistant en mode silencieux. Le créateur du fichier d'options peut ainsi
# spécifier les paramètres de l'assistant sans avoir à exécuter l'assistant
# en mode graphique ou console. Pour utiliser ce fichier d'options en vue d'une
# exécution en mode silencieux, utilisez les arguments de ligne de commande
# suivants lorsque vous exécutez l'assistant :
#
#    -options "D:\installImage\WXS_Client\wxssetup.response" -silent
#
# Notez que le nom de fichier de réponses complet doit être utilisé.
```

```
#
################################################################################

################################################################################
#
# Acceptation de la licence
#
# Valeurs admises :
#  true - Accepte la licence. Installe le produit.
#  false - Refuse la licence. L'installation n'est pas effectuée.
#
# Si aucune installation n'est effectuée, cela est consigné dans un fichier
# journal temporaire, dans le répertoire temporaire de l'utilisateur.
#
# En spécifiant la valeur "true" pour la propriété silentInstallLicenseAcceptance
# dans ce fichier de réponses, vous certifiez que vous avez vérifié et que vous
# acceptez les dispositions des Conditions Internationales d'Utilisation des
# Logiciels IBM accompagnant ce programme, qui se trouve dans
# CD_ROOT\WXS_Cleint\wxs.client.primary.pak\repository\legal.xs.client\license.xs.
# Si vous les refusez, ne modifiez pas la valeur de cette propriété, ne
# téléchargez pas ce programme, ne l'installez pas, ne le copiez pas
# et n'y accédez pas.
# Retournez  rapidement le programme et votre autorisation d'utilisation du
# logiciel au tiers auprès duquel vous l'avez acquis afin d'en obtenir
# le remboursement.
-OPT silentInstallLicenseAcceptance="false"


################################################################################
# Vérification des prérequis non bloquants
#
# Si vous souhaitez désactiver la vérification des prérequis non bloquants,
# supprimez la mise en commentaire de la ligne ci-après. Cette action indique
# au programme d'installation de poursuivre l'installation et de consigner les
# avertissements bien que la vérification des prérequis ait échoué.
#
#-OPT disableNonBlockingPrereqChecking="true"


################################################################################
#
# Emplacement d'installation
#
# Emplacement d'installation du produit. Spécifiez un répertoire valide dans
# lequel le produit doit être installé. Si le répertoire contient des espaces,
# placez-le entre guillemets, comme indiqué dans l'exemple Windows ci-après.
# Notez que les espaces dans l'emplacement d'installation ne sont pris en charge
# que sur les systèmes d'exploitation Windows. La longueur maximale du chemin
# d'accès est de 60 caractères pour Windows.
#
# Vous trouverez ci-après une liste des emplacements d'installation par défaut
# pour chaque système d'exploitation pris en charge si vous effectuez
# l'installation en tant qu'utilisateur root.  Par défaut, dans ce fichier de
# réponses, l'emplacement d'installation de Windows est utilisé.  Si vous
# souhaitez utiliser l'emplacement d'installation par défaut d'un autre
# système d'exploitation, supprimez la mise en commentaire de l'entrée
# d'emplacement d'installation par défaut (en supprimant le signe '#'), puis
# placez en commentaire (en ajoutant le signe '#') l'entrée de système
# d'exploitation Windows ci-après.
#
# L'emplacement d'installation est utilisé pour déterminer si WebSphere eXtreme
# Scale doit être installé comme déploiement autonome ou s'il doit être intégré
# à une installation WebSphere Application Server existante.
#
# Si l'emplacement spécifié correspond à une installation WebSphere Application
# Server ou WebSphere Network Deployment existante, eXtreme Scale est intégré
# au serveur WebSphere Application Server existant.  Si l'emplacement spécifié
```

```
# correspond à un répertoire nouveau ou vide, WebSphere eXtreme Scale est
# installé comme déploiement autonome.
#
# Remarque : Si l'emplacement d'installation spécifié contient une installation
# de WebSphere eXtreme Scale, WebSphere eXtended Deployment DataGrid ou
# ObjectGrid, l'installation échoue.
#
# Emplacement d'installation par défaut sous AIX :
#
#  -OPT installLocation="/usr/IBM/WebSphere/eXtremeScale"
#
# Emplacement d'installation par défaut sous HP-UX, Solaris ou Linux :
#
#  -OPT installLocation="/opt/IBM/WebSphere/eXtremeScale"
#
#
# Emplacement d'installation par défaut sous Windows :
#
-OPT installLocation="C:\Program Files\IBM\WebSphere\eXtremeScale"

#
# Si vous effectuez l'installation en tant qu'utilisateur autre que root sous
# Unix ou qu'administrateur sous Windows, les emplacements d'installation par
# défaut ci-après sont recommandés. Assurez-vous de disposer des droits
# d'accès  en écriture pour l'emplacement d'installation choisi.
#
# Emplacement d'installation par défaut sous AIX :
#
#  -OPT installLocation="<répertoire de base de
#      l'utilisateur>/IBM/WebSphere/eXtremeScale"
#
# Emplacement d'installation par défaut sous HP-UX, Solaris ou Linux :
#
#  -OPT installLocation="<répertoire de base de l'utilisateur>
#  /IBM/WebSphere/eXtremeScale"
#
# Emplacement d'installation par défaut sous Windows :
#
# -OPT installLocation="C:\IBM\WebSphere\eXtremeScale"


###############################################################################
#  Liste des profils à étendre
#
# Spécifiez les profils existants à étendre ou placez en commentaire la ligne
# permettant d'étendre tous les profils existants détectés par l'installation.
#
# Pour spécifier plusieurs profils, utilisez une virgule afin de séparer les
# différents noms de profil.
# Par exemple, "AppSrv01,Dmgr01,Custom01". La liste ne doit pas contenir
# d'espace.

-OPT profileAugmentList=""


###############################################################################
# Contrôle du traçage
#
# Le format de la sortie de trace peut être contrôlé via l'option
# -OPT traceFormat=ALL
#
# Les choix de format sont 'text' et 'XML'. Par défaut, ces deux formats seront
# générés, dans deux fichiers de trace différents.
#
# Si un seul format est requis, utilisez l'option traceFormat pour le spécifier,
# comme suit :
#
```

```
# Valeurs admises :
#
#  text -  Les lignes du fichier de trace sont au format texte pour une meilleure
#          lisibilité.
#  XML  -  Les lignes du fichier de trace se trouveront au format XML
#          de consignation Java standard qui peut être affiché à l'aide
#          de tout éditeur de texte ou éditeur XML ou à l'aide
#          de l'outil Chainsaw d'Apache, à l'adresse URL suivante :
#          (http://logging.apache.org/log4j/docs/chainsaw.html).
#
# La quantité d'informations de trace capturées peut être contrôlée à l'aide de
# l'option suivante :
# -OPT traceLevel=INFO
#
# Valeurs admises :
#
# Niveau    Niveau
# de trace  numérique  Description
# -------  ---------  ------------------------------------------------------
# OFF         0       Aucun fichier de trace n'est généré
# SEVERE      1       Seules les erreurs graves sont consignées
#                     dans le fichier de trace
# WARNING     2       Les messages relatifs aux exceptions non fatales et
#                     aux avertissements sont ajoutés au fichier de trace
# INFO        3       Les messages d'information sont ajoutés
#                     au fichier de trace (niveau de trace par défaut)
# CONFIG      4       Les messages liés à la configuration sont ajoutés
#                     au fichier de trace
# FINE        5       Traçage des appels de méthode pour les
#                     méthodes publiques
# FINER       6       Traçage des appels de méthode pour les méthodes non
#                     publiques, exceptées les méthodes d'accès get et set
# FINEST      7       Traçage de tous les appels de méthode ; l'entrée/
#                     la sortie de trace inclut les paramètres
#                     et la valeur de retour
```

Création et augmentation de profils pour WebSphere eXtreme Scale

Une fois que vous avez installé le produit, créez des types de profil uniques et étendez les profils existants de WebSphere eXtreme Scale.

Avant de commencer

Installez WebSphere eXtreme Scale. Pour plus d'informations, voir «Intégration de WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client à WebSphere Application Server», à la page 22.

L'extension de profils en vue de leur utilisation avec WebSphere eXtreme Scale est facultative, mais requise dans les cas suivants :

- Pour démarrer automatiquement un service de catalogue ou un conteneur dans un processus WebSphere Application Server. Si vous n'étendez pas les profils des serveurs, les serveurs ne peuvent être démarrés qu'à l'aide d'un programme, à l'aide de l'API ServerFactory, ou comme processus distincts, à l'aide des scripts startOgServer.
- Pour utiliser l'infrastructure PMI (Performance Monitoring Infrastructure) afin de surveiller les mesures de WebSphere eXtreme Scale.
- Pour afficher la version de WebSphere eXtreme Scale dans la console d'administration de WebSphere Application Server.

Pourquoi et quand exécuter cette tâche

Exécution dans WebSphere Application Server Version 6.0.2

Si votre environnement contient WebSphere Application Server Version 6.0.2, utilisez la commande wasprofile pour créer ou étendre des profils pour WebSphere eXtreme Scale, comme illustré dans l'exemple suivant :

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

Pour plus d'informations, voir la commande wasprofile dans le Centre de documentation de WebSphere Application Server.

Exécution dans WebSphere Application Server Version 6.1 ou 7.0

Si votre environnement contient WebSphere Application Server Version 6.1 ou 7.0, vous pouvez utiliser le plug-in de l'outil de gestion de profil ou la commande manageprofiles pour créer et étendre des profils.

Que faire ensuite

Suivant la tâche que vous choisissez d'effectuer, lancez la console Premiers pas pour obtenir de l'aide lors de la configuration et du test de l'environnement de votre produit. La console Premiers pas se trouve dans le répertoire <*racine_install*>\firststeps\wxs\firststeps.bat. Vous pouvez également créer ou étendre des profils supplémentaires en répétant l'une des tâches précédentes.

Utilisation de l'interface graphique pour créer des profils

Utilisez l'interface graphique, offerte par le plug-in de l'outil de gestion de profil, afin de créer des profils pour WebSphere eXtreme Scale. Un profil est un ensemble de fichiers qui définissent l'environnement d'exécution.

Avant de commencer

Remarque : Si vous exécutez WebSphere Application Server Version 6.0.2 ou WebSphere Application Server Network Deployment Version 6.0.2, vous devez utiliser la commande wasprofile pour créer ou étendre un profil pour WebSphere eXtreme Scale, comme illustré dans l'exemple suivant :

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

Pour plus d'informations, voir la commande wasprofile dans le Centre de documentation de WebSphere Application Server Version 6.0.

Pourquoi et quand exécuter cette tâche

Pour utiliser les fonctions du produit, le plug-in de l'outil de gestion de profil permet à l'interface graphique de vous aider à configurer des profils, tels qu'un profil WebSphere Application Server, un profil de gestionnaire de déploiement, un profil de cellule et un profil personnalisé.

Procédure

Utilisez l'interface graphique de l'outil de gestion des profils pour créer des profils. Choisissez l'une des options suivantes pour démarrer l'assistant :

- Sélectionnez **Outil de gestion de profil** dans la console Premiers pas.

- Accédez à l'outil de gestion de profil à partir du menu **Démarrer**.
- Exécutez le script `./pmt.sh|bat` à partir du répertoire *racine_install*`/bin/ProfileManagement`.

Que faire ensuite

Vous pouvez créer d'autres profils ou étendre des profils existants. Pour redémarrer l'outil de gestion de profil, exécutez la commande ./pmt.sh|bat à partir du répertoire *racine_install*`/bin/ProfileManagement` ou sélectionnez **Outil de gestion de profil** dans la console Premiers pas.

Démarrez un service de catalogue, démarrez des conteneurs et configurez les ports TCP dans votre environnement WebSphere Application Server. Pour plus d'informations, voir «Administration de WebSphere eXtreme Scale avec WebSphere Application Server», à la page 344.

Utilisation de l'interface graphique pour étendre des profils

Après avoir installé le produit, vous pouvez étendre un profil existant pour le rendre compatible avec WebSphere eXtreme Scale.

Avant de commencer

Remarque : Si vous exécutez WebSphere Application Server Version 6.0.2 ou WebSphere Application Server Network Deployment Version 6.0.2, vous devez utiliser la commande wasprofile pour créer ou étendre un profil pour WebSphere eXtreme Scale, comme illustré dans l'exemple suivant :

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

Pour plus d'informations, voir la commande wasprofile dans le Centre de documentation de WebSphere Application Server.

Pourquoi et quand exécuter cette tâche

Lorsque vous étendez un profil existant, vous modifiez le profil en appliquant un modèle d'extension spécifique à un produit. Par exemple, les serveurs WebSphere eXtreme Scale ne démarrent pas automatiquement à moins que le profil du serveur ne soit étendu avec le modèle xs_augment.

- Etendez le profil avec le modèle xs_augment si vous avez installé le client eXtreme Scale ou le client et le serveur.
- N'étendez le profil avec le modèle pf_augment que si vous avez installé la fonction de partitionnement.
- Appliquez les deux modèles si votre environnement contient le client eXtreme Scale et l'utilitaire de partitionnement.

Procédure

Utilisez l'interface graphique de l'outil de gestion des profils pour étendre les profils pour eXtreme Scale. Choisissez l'une des options suivantes pour démarrer l'assistant :

- Sélectionnez **Outil de gestion de profil** dans la console Premiers pas.
- Accédez à l'outil de gestion de profil à partir du menu **Démarrer**.
- Exécutez le script ./pmt.sh|bat à partir du répertoire *racine_install*`/bin/ProfileManagement`.

Que faire ensuite

Vous pouvez étendre d'autres profils. Pour redémarrer l'outil de gestion de profil, exécutez la commande ./pmt.sh|bat à partir du répertoire *racine_install*/bin/ProfileManagement ou sélectionnez **Outil de gestion de profil** dans la console Premiers pas.

Démarrez un service de catalogue, démarrez des conteneurs et configurez les ports TCP dans votre environnement WebSphere Application Server. Pour plus d'informations, voir «Administration de WebSphere eXtreme Scale avec WebSphere Application Server», à la page 344.

Commande manageprofiles

Vous pouvez utiliser l'utilitaire manageprofiles pour créer des profils à l'aide du modèle WebSphere eXtreme Scale ou étendre et réduire des profils de serveur d'applications existants à l'aide des modèle d'extension de eXtreme Scale. Pour utiliser les fonctions du produit, votre environnement doit contenir au moins un profil étendu pour le produit.

- Pour pouvoir créer et étendre des profils, vous devez installer eXtreme Scale. Pour plus d'informations, voir «Intégration de WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client à WebSphere Application Server», à la page 22.
- Si votre environnement contient WebSphere Application Server Version 6.0.2, vous devez utiliser la commande wasprofile pour créer et étendre des profils pour eXtreme Scale, comme illustré dans l'exemple suivant :

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

  Pour plus d'informations, voir la commande wasprofile dans le Centre de documentation de WebSphere Application Server.

Rôle

La commande manageprofiles crée l'environnement d'exécution d'un processus de produit dans un ensemble de fichiers appelé profil. Le profil définit l'environnement d'exécution. Vous pouvez effectuer les actions suivantes à l'aide de la commande manageprofiles :

- Création et extension d'un profil de gestionnaire de déploiement
- Création et extension d'un profil personnalisé
- Création et extension d'un profil de serveur d'applications autonome
- Création et extension d'un profil de cellule
- Réduction de tout type de profil

Lorsque vous étendez un profil existant, vous modifiez le profil en appliquant un modèle d'extension spécifique à un produit.

- Etendez le profil avec le modèle xs_augment si vous avez installé le client eXtreme Scale ou le client et le serveur.
- Etendez le profil avec le modèle pf_augment si vous n'avez installé que l'utilitaire de partitionnement.
- Appliquez les deux modèles si votre environnement contient le client eXtreme Scale et l'utilitaire de partitionnement.

Emplacement

Le fichier de commandes se trouve dans le répertoire *racine_install*/bin.

Syntaxe

Pour obtenir une aide détaillée, utilisez le paramètre **-help** :

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

Dans les sections ci-après, chaque tâche que vous pouvez effectuer à l'aide de la commande manageprofiles est décrite, avec une liste des paramètres requis. Pour des détails sur les paramètres facultatifs à spécifier pour chaque tâche, reportez-vous à la commande manageprofiles, dans le Centre de documentation de WebSphere Application Server.

Création d'un profil de gestionnaire de déploiement

Vous pouvez utiliser la commande manageprofiles pour créer un profil de gestionnaire de déploiement. Le gestionnaire de déploiement administre les serveurs d'applications fédérés dans la cellule.

Paramètres

-create
Crée un profil. (Obligatoire)

-templatePath *chemin_modèle*
Indique le chemin du modèle. (Obligatoire)

Utilisez le format suivant :

```
-templatePath racine_install/profileTemplates/type_modèle/dmgr
```

où *type_modèle* est `xs_augment` ou `pf_augment`.

Exemple

- Utilisation du modèle xs_augment :

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

- Utilisation du modèle pf_augment :

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

Création d'un profil personnalisé.

Vous pouvez utiliser la commande manageprofiles pour créer un profil personnalisé. Un profil personnalisé est un noeud vide que vous personnalisez via le gestionnaire de déploiement pour inclure des serveurs d'applications, des clusters ou d'autres processus Java.

Paramètres

-create
Crée un profil. (Obligatoire)

-templatePath *chemin_modèle*
Indique le chemin du modèle. (Obligatoire)

Utilisez le format suivant :

```
-templatePath racine_install/profileTemplates/type_modèle/managed
```

où *type_modèle* est `xs_augment` ou `pf_augment`.

Exemple

- Utilisation du modèle xs_augment :

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

- Utilisation du modèle pf_augment :

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

Création d'un profil de serveur d'applications autonome

Vous pouvez utiliser la commande manageprofiles pour créer un profil de serveur d'applications autonome.

Paramètres

-create
Crée un profil. (Obligatoire)

-templatePath *chemin_modèle*
Indique le chemin du modèle. (Obligatoire)

Utilisez le format suivant :

```
-templatePath racine_install/profileTemplates/type_modèle/default
```

où *type_modèle* est `xs_augment` ou `pf_augment`.

Exemple

- Utilisation du modèle xs_augment :

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

- Utilisation du modèle pf_augment :

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

Création d'un profil de cellule

Vous pouvez utiliser la commande manageprofiles pour créer un profil de serveur, qui comprend un gestionnaire de déploiement et un serveur d'applications.

Paramètres

Indiquez les paramètres suivants dans le modèle de gestionnaire de déploiement :

-create
Crée un profil. (Obligatoire)

-templatePath *chemin_modèle*
Indique le chemin du modèle. (Obligatoire)

Utilisez le format suivant :

```
-templatePath racine_install/profileTemplates/type_modèle/cell/dmgr
```

où *type_modèle* est `xs_augment` ou `pf_augment`.

Indiquez les paramètres suivants avec le modèle de serveur d'applications :

-create
Crée un profil. (Obligatoire)

-templatePath *chemin_modèle*
Indique le chemin du modèle. (Obligatoire)

Utilisez le format suivant :

```
-templatePath racine_install/profileTemplates/type_modèle/cell/default
```

où *type_modèle* est `xs_augment` ou `pf_augment`.

Exemple

- Utilisation du modèle xs_augment :

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

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

- Utilisation du modèle pf_augment :

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

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

Extension d'un profil de gestionnaire de déploiement

Vous pouvez utiliser la commande manageprofiles pour étendre un profil de gestionnaire de déploiement.

Paramètres

-augment
Etend le profil existant. (Obligatoire)

-profileName
Spécifie le nom du profil. (Obligatoire)

-templatePath *chemin_modèle*
Spécifie le chemin des fichiers de modèle qui se trouvent dans le répertoire racine d'installation. (Obligatoire)

Utilisez le format suivant :

```
-templatePath racine_install/profileTemplates/type_modèle/dmgr
```

où *type_modèle* est `xs_augment` ou `pf_augment`.

Exemple

- Utilisation du modèle xs_augment :

```
./manageprofile.sh|bat -augment -profileName profile01

-templatePath racine_install
/profileTemplates/xs_augment/dmgr
```

- Utilisation du modèle pf_augment :

```
./manageprofile.sh|bat -augment -profileName profile01

-templatePath racine_install
/profileTemplates/pf_augment/dmgr
```

Extension d'un profil personnalisé

Vous pouvez utiliser la commande manageprofiles pour étendre un profil personnalisé.

Paramètres

-augment
Etend le profil existant. (Obligatoire)

-profileName
Spécifie le nom du profil. (Obligatoire)

-templatePath *chemin_modèle*
Spécifie le chemin des fichiers de modèle qui se trouvent dans le répertoire racine d'installation. (Obligatoire)

Utilisez le format suivant :

```
-templatePath racine_install/profileTemplates/type_modèle/managed
```

où *type_modèle* est `xs_augment` ou `pf_augment`.

Exemple

- Utilisation du modèle xs_augment :

```
./manageprofile.sh|bat -augment -profileName profile01

-templatePath racine_install
   /profileTemplates/xs_augment/managed
```

- Utilisation du modèle pf_augment :

```
./manageprofile.sh|bat -augment -profileName profile01

-templatePath racine_install
   /profileTemplates/pf_augment/managed
```

Extension d'un profil de serveur d'applications autonome

Vous pouvez utiliser la commande manageprofiles pour étendre un profil de serveur d'applications autonome.

Paramètres

-augment
Etend le profil existant. (Obligatoire)

-profileName
Spécifie le nom du profil. (Obligatoire)

-templatePath *chemin_modèle*
Spécifie le chemin des fichiers de modèle qui se trouvent dans le répertoire racine d'installation. (Obligatoire)

Utilisez le format suivant :

```
-templatePath racine_install/profileTemplates/type_modèle/default
```

où *type_modèle* est `xs_augment` ou `pf_augment`.

Exemple

- Utilisation du modèle xs_augment :

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

- Utilisation du modèle pf_augment :

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

Extension d'un profil de cellule

Vous pouvez utiliser la commande manageprofiles pour étendre un profil de cellule.

Paramètres

Indiquez les paramètres suivants pour le profil de gestionnaire de déploiement :

-augment
Etend le profil existant. (Obligatoire)

-profileName
Spécifie le nom du profil. (Obligatoire)

-templatePath *chemin_modèle*
Spécifie le chemin des fichiers de modèle qui se trouvent dans le répertoire racine d'installation. (Obligatoire)

Utilisez le format suivant :

```
-templatePath racine_install/profileTemplates/type_modèle/cell/dmgr
```

où *type_modèle* est `xs_augment` ou `pf_augment`.

Indiquez les paramètres suivants pour le profil de serveur d'applications :

-augment
Etend le profil existant. (Obligatoire)

-profileName
Spécifie le nom du profil. (Obligatoire)

-templatePath *chemin_modèle*
Spécifie le chemin des fichiers de modèle qui se trouvent dans le répertoire racine d'installation. (Obligatoire)

Utilisez le format suivant :

```
-templatePath racine_install/profileTemplates/type_modèle/cell/default
```

où *type_modèle* est `xs_augment` ou `pf_augment`.

Exemple

- Utilisation du modèle xs_augment :

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

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

- Utilisation du modèle pf_augment :

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

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

Réduction d'un profil

Pour réduire un profil, spécifiez le paramètre **-ignoreStack** avec le paramètre **-templatePath**, en plus des paramètres **-unaugment** et **-profileName** requis.

Paramètres

-unaugment
Réduit un profil précédemment étendu. (Obligatoire)

-profileName
Spécifie le nom du profil. Le paramètre est généré par défaut si aucune valeur n'est spécifiée. (Obligatoire)

-templatePath *chemin_modèle*
Spécifie le chemin des fichiers de modèle qui se trouvent dans le répertoire racine d'installation (facultatif).

Utilisez le format suivant :

```
-templatePath racine_install/profileTemplates/type_modèle/type_profil
```

où *type_modèle* est `xs_augment` ou `pf_augment` et *type_profil* correspond à l'un des quatre types de profil suivants :

- `dmgr` : profil du gestionnaire de déploiement
- `managed` : profil personnalisé
- `default` : profil de serveur d'applications autonome
- `cell` : profil de cellule

-ignoreStack
Utilisé avec le paramètre **-templatePath** pour réduire un profil particulier qui a été étendu (facultatif).

Exemple

- Utilisation du modèle xs_augment :

```
./manageprofile.sh|bat -unaugment -profileName profile01 -ignoreStack
-templatePath racine_install/profileTemplates/xs_augment/type_profil
```

- Utilisation du modèle pf_augment :

```
./manageprofile.sh|bat -unaugment -profileName profile01 -ignoreStack
-templatePath racine_install/profileTemplates/pf_augment/type_profil
```

Profils non root

Vous pouvez octroyer à l'utilisateur non root des autorisations pour des fichiers et répertoires afin de permettre à cet utilisateur de créer un profil pour le produit. L'utilisateur non root peut également augmenter un profil qui a été créé par un utilisateur root, par un autre utilisateur non root ou par lui-même.

Dans un environnement WebSphere Application Server, les autorisations de création et d'utilisation de profils des utilisateurs non root (non administrateurs) sont limitées. Dans le plug-in de l'outil de gestion de profil, les noms et les valeurs de port uniques sont désactivés pour les utilisateurs non root. Ces derniers doivent modifier dans l'outil de gestion de profilles valeurs par défaut des zones de nom du profil, de nom du noeud, de nom de la cellule et d'affectations des ports. Pensez à affecter aux utilisateurs non root une plage de valeurs pour chacune de ces zones. Vous pouvez attribuer la responsabilité à des utilisateurs non root d'adhérer aux plages de valeurs adéquates et de maintenir l'intégrité de leurs propres définitions.

Par *responsable de l'installation*, l'on entend soit l'utilisateur root, soit des utilisateurs non root. En tant que responsable de l'installation, vous pouvez octroyer aux utilisateurs non root des autorisations de création de profils et d'établissement de leurs propres environnements de produit. Par exemple, un utilisateur non root pourra créer un environnement de produit afin de tester un déploiement

d'application avec un profil dont il est le propriétaire. Les autorisations de création de profils accordées aux utilisateurs non root comprennent les éléments suivants :

- création d'un profil et attribution de la propriété du répertoire du profil à un utilisateur non root pour lui permettre de démarrer WebSphere Application Server pour un profil spécifique
- octroi de droits d'accès en écriture aux fichiers et répertoires appropriés à un utilisateur non root pour lui permettre de créer le profil. Cette tâche permet de créer un groupe d'utilisateurs autorisés à créer des profils ou d'accorder à des utilisateurs individuels la possibilité de créer des profils
- installation de packages de maintenance pour le produit, ce qui inclut les services requis pour les profils existants appartenant à un utilisateur non root. En tant que responsable de l'installation, c'est vous le propriétaire de tous les fichiers créés par ce package de maintenance

Pour plus d'informations sur la création de profils pour les utilisateurs non root, reportez-vous à la rubrique Création de profils pour les utilisateurs non root .

En tant que responsable de l'installation, vous pouvez également octroyer aux utilisateurs non root des autorisations d'extension de profils. Par exemple, un utilisateur non root peut étendre un profil créé par un responsable d'installation ou par lui-même. Suivez les extensions de profils réalisées par des utilisateurs non root de WebSphere Application Server Network Deployment.

Toutefois, lorsqu'un utilisateur non root étend un profil créé par le responsable de l'installation, il n'a pas besoin de créer auparavant les fichiers indiqué ci-après. Ces fichiers ont été créés en même temps que le profil.

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

Lorsqu'un utilisateur non root étend un profil qu'il crée, il doit modifier les autorisations relatives aux documents situés dans les modèles de profils eXtreme Scale.

Avertissement : Vous pouvez également utiliser un profil non root (non administrateur) pour WebSphere eXtreme Scale dans un environnement autonome, extérieur à WebSphere Application Server. Vous devez remplacer le propriétaire du répertoire ObjectGrid par le profil non root. Vous pouvez ensuite ouvrir une session avec ce profil non root et utiliser eXtreme Scale comme vous le feriez avec un profil root (administrateur).

Installation de WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client en mode silencieux

Utilisez un fichier de réponses complet, que vous configurez spécifiquement en fonction de vos besoins ou transmettez les paramètres à la ligne de commande pour effectuer une installation de WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client en mode silencieux.

Avant de commencer

- Arrêtez tous les processus en cours d'exécution dans votre environnement WebSphere Application Server ou WebSphere Application Server Network Deployment. Pour en savoir plus sur les commmandes stopManager, stopNode et stopServer, voir Using wsadmin scripting command line tools.

ATTENTION :
Vérifiez que les processus en cours d'exécution sont arrêtés. Si les processus en cours d'exécution ne sont pas arrêtés, l'installation se poursuit et crée des résultats imprévisibles, ce qui la laisse dans un état indéterminé sur certaines plateformes.

- Vérifiez que le répertoire d'installation cible est vide ou qu'il n'existe pas.

  Important : Si une version précédente d'WebSphere eXtreme Scale ou le composant ObjectGrid se trouve dans le répertoire que vous spécifiez pour installer la version 7.1, le produit n'est pas installé. Par exemple, il peut exister un dossier `<racine_install_wxs>/ObjectGrid`. Vous pouvez sélectionner un autre répertoire d'installation ou annuler l'installation. Ensuite, désinstallez l'installation précédente et exécutez de nouveau l'assistant.

Pourquoi et quand exécuter cette tâche

Une installation en mode silencieux utilise le même programme d'installation que l'interface graphique. Toutefois, au lieu d'afficher une interface d'assistant, elle lit toutes vos réponses dans un fichier que vous personnalisez ou dans les paramètres que vous transmettez à la ligne de commande. Consultez un exemple de «Fichier `wxssetup.response.txt`», à la page 35, qui inclut une description de chaque option.

Procédure

1. Facultatif : Si vous choisissez d'installer WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client à l'aide d'un fichier de réponses, personnalisez d'abord le fichier `wxssetup.response.txt`.

   A faire : Vous devez spécifier le nom complet du fichier de réponses. Si vous spécifiez le chemin relatif, l'installation échoue sans message d'erreur.

   a. Effectuez une copie du fichier de réponses à personnaliser.

      Pour l'installation intégrale de WebSphere eXtreme Scale, copiez le fichier de réponses du DVD du produit vers votre unité de disque.

      Pour WebSphere eXtreme Scale Client, décompressez le fichier zip de WebSphere eXtreme Scale Client sur votre disque dur et recherchez le fichier de réponses.

   b. Ouvrez et éditez le fichier de réponses dans l'éditeur de texte de votre choix. L'exemple de fichier de réponses précédent fournit des détails sur la manière de spécifier chacun des paramètres. Vous devez spécifier les paramètres suivants :

      - Contrat de licence
      - Répertoire d'installation

      Conseil : Lorsque vous installez WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client dans un environnement WebSphere Application Server, le programme d'installation utilise le répertoire d'installation pour déterminer où l'instance WebSphere Application Server est installée. Si vous effectuez l'installation sur un noeud qui contient plusieurs instances WebSphere Application Server, définissez précisément votre emplacement.

   c. Exécutez le script suivant pour démarrer l'installation.

      Pour l'installation intégrale de WebSphere eXtreme Scale :

      ```
      ./install.sh|bat -options C:/chemin_unité/fichier_réponse.txt -silent
      ```

      Pour l'installation de WebSphere eXtreme Scale Client :

      ```
      ./WXS_Client/install.sh|bat -options C:/chemin_unité/fichier_réponses.txt -silent
      ```

Vous pouvez également utiliser le fichier de réponses lorsque vous exécutez une installation à l'aide de l'interface graphique. Vous pouvez utiliser le fichier de réponses avec une installation à l'aide de l'interface graphique pour déboguer les problèmes masqués par l'installation en mode silencieux. Lorsque vous spécifiez le fichier `wxssetup.response` pour des installations à l'aide de l'interface graphique ou en mode silencieux, vous devez utiliser le chemin qualifié complet. Exécutez le script suivant pour exécuter l'installation à l'aide de l'interface graphique avec votre fichier de réponses :

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

2. Facultatif : Si vous choisissez d'installer eXtreme Scale en transmettant certains paramètres à la ligne de commande, exécutez le script suivant pour démarrer l'installation :

   Pour l'installation intégrale de WebSphere eXtreme Scale :

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

   Pour l'installation de WebSphere eXtreme Scale Client :

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

Paramètres d'installation

Spécifiez des paramètres sur la ligne de commande pour personnaliser et configurer l'installation de votre produit.

Remarque : Vous devez spécifier le nom complet du fichier de réponses. Si vous spécifiez le chemin relatif, l'installation échoue sans message d'erreur.

Paramètres

Vous pouvez transmettre les paramètres suivants lors d'une installation du produit à l'aide de la ligne de commande ou du fichier d'options :

-silent
Supprime l'interface graphique. Spécifiez le paramètre **-options** pour indiquer que le programme d'installation effectue l'installation en fonction d'un fichier d'options personnalisé. Si vous ne spécifiez pas le paramètre **-options**, les valeurs par défaut sont utilisées.

Exemple de syntaxe

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

-options *nom_chemin/nom_fichier*
Indique un fichier d'options utilisé par le programme d'installation pour effectuer une installation en mode silencieux. Les propriétés sur la ligne de commande sont prioritaires.

Exemple de syntaxe

```
./install.sh|bat -options c:/nom_chemin/fichier_options.txt
```

-log # !file_name @*type_événement*
Génère un fichier journal d'installation qui consigne les types d'événement suivants :

- `err`

- wrn
- msg1
- msg2
- dbg
- ALL

Exemple de syntaxe

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

-is:log *nom_chemin/nom_fichier*
Crée un fichier journal qui contient les recherches JVM (Java Virtual Machine) du programme d'installation lors de la tentative de démarrage de l'interface graphique. Le fichier journal n'est pas créé s'il n'est pas spécifié.

Exemple de syntaxe

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

-is:javaconsole
Affiche une fenêtre de console lors de la procédure d'installation.

Exemple de syntaxe

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

-is:silent
Supprime la fenêtre d'initialisation Java affichée au démarrage du programme d'installation.

Exemple de syntaxe

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

-is:tempdir *nom_chemin*
Indique le répertoire temporaire utilisé par le programme d'installation lors de l'installation.

Exemple de syntaxe

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

Utilisation du programme d'installation de mises à jour pour installer des modules de maintenance

Utilisez IBM Update Installer pour mettre à jour votre environnement WebSphere eXtreme Scale avec divers types de maintenance, tels que les correctifs temporaires, les groupes de correctifs et les groupes de mises à jour.

Pourquoi et quand exécuter cette tâche

Utilisez IBM Update Installer pour installer et appliquer plusieurs types de modules de maintenance pour WebSphere eXtreme Scale. Ce programme d'installation de mises à jour étant soumis à des opérations de maintenance régulières, veuillez utiliser sa dernière version.

Procédure

1. Arrêtez tous les processus en cours d'exécution dans votre environnement.
   - Pour savoir comment arrêter tous les processus en cours d'exécution dans votre environnement autonome eXtreme Scale, voir «Arrêt de serveurs eXtreme Scale autonomes», à la page 339 les explications sur le démarrage et l'arrêt de serveurs.

- Pour arrêter tous les processus en cours d'exécution dans votre environnement WebSphere Application Server, voir le document sur les utilitaires de ligne de commande.

2. Téléchargez la dernière version du programme d'installation de mises à jour. Voir Correctifs recommandés pour plus d'informations.
3. Installez le programme d'installation de mises à jour. Voir l'installation Update Installer for WebSphere Software dans WebSphere Application Server le Centre de documentation pour plus d'informations.
4. Téléchargez dans le répertoire *updi_root*/maintenance les modules de maintenance à installer. Voir Site de support pour plus d'informations.
5. Utilisez le programme d'installation de mises à jour pour installer le correctif temporaire, le groupe de correctifs ou le groupe de mises à jour. Vous pouvez installer le module de maintenance en exécutant l'interface graphique ou le programme d'installation de mises à jour en mode silencieux.

   Exécutez la commande suivante à partir du répertoire *updi_root* pour démarrer l'interface graphique :

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

   Exécutez la commande suivante à partir du répertoire *updi_root* pour lancer le programme d'installation de mises à jour en mode silencieux :

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

   En cas d'échec du processus d'installation, consultez le fichier journal temporaire, situé dans le répertoire *updi_root*/logs/update/tmp. Le programme d'installation de mises à jour crée le répertoire *install_root*/logs/update/*maintenance_package*.install qui contient les journaux d'installation.

Désinstallation de WebSphere eXtreme Scale

Pour supprimer WebSphere eXtreme Scale de votre environnement, vous pouvez utiliser l'assistant ou procéder à une désinstallation en mode silencieux.

Avant de commencer

Avertissement : Le programme de désinstallation supprime tous les fichiers binaires et la maintenance, telle que les groupes de correctifs et les correctifs temporaires, en même temps.

Procédure

1. Arrêtez tous les processus exécutant eXtreme Scale.

   ATTENTION :
   Assurez-vous que tous les processus en cours d'exécution sont bien arrêtés. Si les processus ne sont pas arrêtés, la désinstallation s'effectue, créant des résultats imprévisibles et laissant sur quelques plateformes la désinstallation dans un état indéterminé.

   - Si vous avez installé un serveur eXtreme Scale autonome, lisez la rubrique sur l'arrêt des serveurs autonomes pour arrêter les processus.
   - Si vous avez installé eXtreme Scale avec une installation existante de WebSphere Application Server, lisez le document sur les utilitaires de ligne de commande pour savoir comment arrêter les processus de WebSphere Application Server.

2. Désinstallez le produit. La désinstallation peut s'effectuer dans une interface graphique ou en mode silencieux.

   Remarque : Lorsqu'on spécifie le fichier de réponses `wxssetup.response` pour une désinstallation ou pour des installations dans l'interface graphique ou en mode silencieux, le chemin qualifié complet du fichier doit être spécifié. Le fichier de réponses n'est pas obligatoire dans le cas d'une désinstallation à partir de l'interface graphique.

   - **Pour exécuter la désinstallation à partir de l'interface graphique :**
     - Linux UNIX `<rép_install>/uninstall_wxs/uninstall`
     - Windows `<rép_install>\uninstall_wxs\uninstall.exe`

     Pour exécuter la désinstallation à partir de l'interface graphique et avec le fichier `wxssetup.response`, utilisez l'une des commandes suivantes :

     - Linux UNIX

       ```
       <rép_install>/uninstall_wxs/uninstall -options
       <chemin_complet_install_requis>/wxssetup.response
       ```

     - Windows

       ```
       <rép_install>\uninstall_wxs\uninstall.exe -options
       <chemin_complet_install_requis>\wxssetup.response
       ```

   - **Pour exécuter la désinstallation en mode silencieux avec le script `wxssetup.response` de fichier de réponses :**
     - Linux UNIX

       ```
       <rép_install>/uninstall_wxs/uninstall -options
       <chemin_complet_install_requis>/wxssetup.response -silent
       ```

     - Windows

       ```
       <rép_install>\uninstall_wxs\uninstall.exe -options
       <chemin_complet_install_requis>\wxssetup.response -silent
       ```

Résultats

Vous avez supprimé eXtreme Scale de votre environnement.

Chapitre 4. Personnalisation de WebSphere eXtreme Scale for z/OS

WebSphere Customization Tools vous permet de générer et d'exécuter des travaux personnalisés pour personnaliser WebSphere eXtreme Scale for z/OS.

Avant de commencer

- Vérifiez que votre système contient bien le niveau le plus récent de WebSphere Application Server Network Deployment :
  - Si vous exécutez la version 6.1, votre système doit contenir au moins le groupe de correctifs 31. Voir Installation de l'environnement de traitement d'applications Version 6.1 pour plus d'informations.
  - Si vous exécutez la version 7.0, votre système doit contenir au moins le groupe de correctifs 9. Voir Installation de l'environnement de traitement d'applications Version 7.0 pour plus d'informations.
- Installez WebSphere eXtreme Scale for z/OS. Pour plus d'informations, voir *le répertoire du programme WebSphere eXtreme Scale WebSphere eXtreme Scale* dans la page des bibliothèques.

Pourquoi et quand exécuter cette tâche

WebSphere Customization Tools vous permet de générer des définitions de personnalisation, ainsi que de télécharger et d'exécuter des travaux personnalisés pour personnaliser WebSphere eXtreme Scale for z/OS. Pour plus d'informations, voir les rubriques suivantes :

Procédure

- «Installation de WebSphere»
- «Génération de définitions de personnalisation», à la page 61
- «Téléchargement et exécution de travaux personnalisés», à la page 62

Installation de WebSphere

Installez WebSphere Customization Tools Version 7.0.0.6 ou ultérieure pour personnaliser votre environnement WebSphere eXtreme Scale for z/OS.

Avant de commencer

Installez WebSphere eXtreme Scale for z/OS. Pour plus d'informations, voir *le répertoire du programme WebSphere eXtreme Scale* dans la page des bibliothèques.

Pourquoi et quand exécuter cette tâche

WebSphere Customization Tools est un outil graphique utilisable sur les postes de travail qui sert à créer des travaux personnalisés générant des environnements d'exécution WebSphere eXtreme Scale for z/OS.

Procédure

1. Utilisez FTP pour copier les fichiers d'extension `xs.wct` et `xspf.wct` de votre système z/OS vers le poste de travail sur lequel vous installez WebSphere

Customization Tools. Les fichiers d'extension se trouvent dans le répertoire `/usr/lpp/zWebSphereXS/util/V7R1/WCT` sur votre système z/OS.

2. Téléchargez et installez WebSphere Customization Tools Version 7.0.0.6 ou version ultérieure depuis le site Web approprié :
   - Windows WebSphere Customization Tools for Windows®
   - Linux WebSphere Customization Tools for Linux®
3. Téléchargez le fichier `xs.wct` vers l'application WebSphere Customization Tools.
   a. Démarrez l'application WebSphere Customization Tools sur le poste de travail.
   b. Cliquez sur **Aide** → **Mises à jour de logiciels** → **Installer l'extension**.
   c. Dans le panneau Emplacements d'extension de WebSphere Customization Tools, cliquez sur **Installer un nouvel emplacement d'extension**.
   d. Dans le panneau Fichier archive source, cliquez sur **Parcourir**, accédez au répertoire dans lequel vous avez copié le fichier `xs.wct` à l'étape 1 et cliquez sur **Ouvrir**.
   e. Cliquez sur **Suivant** dans le panneau Récapitulatif.

      Remarque : Le panneau Installation réussie s'affiche. Avant de cliquer sur **Terminer**, vous devez copier et enregistrer les données depuis la zone réservée à l'emplacement :

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

   f. Dans le panneau Configuration du produit, cliquez sur **Ajouter un emplacement d'extension**. Collez les données copiées à l'étape précédente dans la zone Emplacement et cliquez sur **OK**.
   g. Cliquez sur **Oui** pour redémarrer WebSphere Customization Tools.
4. Téléchargez le fichier `xspf.wct` vers l'application WebSphere Customization Tools.
   a. Cliquez sur **Aide** → **Mises à jour de logiciels** → **Installer l'extension**.
   b. Dans le panneau Emplacements d'extension de WebSphere Customization Tools, cliquez sur **Installer un nouvel emplacement d'extension**.
   c. Dans le panneau Fichier archive source, cliquez sur **Parcourir**, accédez au répertoire dans lequel vous avez copié le fichier `xspf.wct` à l'étape 1 et cliquez sur **Ouvrir**.
   d. Cliquez sur **Suivant** dans le panneau Récapitulatif.

      Remarque : Le panneau Installation réussie s'affiche. Avant de cliquer sur **Terminer**, vous devez copier et enregistrer les données depuis la zone réservée à l'emplacement :

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

   e. Dans le panneau Configuration du produit, cliquez sur **Ajouter un emplacement d'extension**. Collez les données copiées à l'étape précédente dans la zone Emplacement et cliquez sur **OK**.
   f. Cliquez sur **Oui** pour redémarrer WebSphere Customization Tools.

Que faire ensuite

Après avoir téléchargé les fichiers d'extension et redémarré WebSphere Customization Tools, vous pouvez utiliser l'outil de gestion de profil pour générer des définitions de personnalisation pour eXtreme Scale for z/OS. Pour plus d'informations, voir «Génération de définitions de personnalisation», à la page 61.

Génération de définitions de personnalisation

Utilisez la fonction Outil de gestion de profil de WebSphere Customization Tools pour générer des définitions de personnalisation et créer des travaux personnalisés pour WebSphere eXtreme Scale for z/OS.

Avant de commencer

Installez WebSphere Customization Tools et téléchargez les fichiers d'extension `xs.wct` et `xspf.wct`. Pour plus d'informations, voir «Installation de WebSphere», à la page 59.

Pourquoi et quand exécuter cette tâche

Vous pouvez générer des définitions de personnalisation à l'aide de l'outil de gestion de profil, fourni avec WebSphere Customization Tools. Une *définition de personnalisation* est un ensemble de fichiers servant à créer des travaux personnalisés pour configurer WebSphere eXtreme Scale for z/OS.

Procédure

1. Démarrez l'outil de gestion de profil.
   - Windows Cliquez sur **Démarrer** → **Programmes** → **IBM WebSphere** → **WebSphere Customization Tools**. Après le démarrage de l'application, cliquez sur l'onglet **Outil de gestion de profil**.
   - Linux Cliquez sur *operating_system_menus* → **IBM WebSphere** → **WebSphere Customization Tools**. Après le démarrage de l'application, cliquez sur l'onglet **Outil de gestion de profil**.
2. Ajoutez un emplacement existant ou créez l'emplacement de la définition de personnalisation à créer. Dans l'onglet **Emplacements de personnalisation**, cliquez sur **Ajouter**. Si vous créez un emplacement, la zone Version référence la version du produit WebSphere Application Server installé sur votre système z/OS.

   Remarque : N'utilisez pas l'emplacement contenant les autres eXtreme Scaledéfinitions de personnalisation.
3. Générez la définition de personnalisation. Dans l'onglet **Définitions de personnalisation**, cliquez sur **Etendre**.
4. Sélectionnez le type d'environnement de définition à créer :
   - Noeud de serveur d'applications autonome
   - Gestionnaire de déploiement
   - Serveur d'applications
   - Noeud (personnalisé) géré
5. Complétez les zones des panneaux. Spécifiez les valeurs des paramètres utilisés pour la création de votre système z/OS.
6. Cliquez sur **Etendre** pour générer la définition de personnalisation.

Que faire ensuite

Téléchargez le travail personnalisé sur votre système z/OS cible. Pour plus d'informations, voir «Téléchargement et exécution de travaux personnalisés», à la page 62.

Téléchargement et exécution de travaux personnalisés

Après avoir généré les définitions de personnalisation, vous pouvez télécharger et exécuter les travaux personnalisés associés aux définitions sur votre système WebSphere eXtreme Scale for z/OS

Avant de commencer

Générez les définitions de personnalisation pour les travaux à télécharger sur votre système z/OS. Pour plus d'informations, voir «Génération de définitions de personnalisation», à la page 61.

Pourquoi et quand exécuter cette tâche

Téléchargez et exécutez les travaux personnalisés que vous avez créés à l'aide de WebSphere Customization Tools de manière à administrer et à surveiller votre environnement WebSphere eXtreme Scale for z/OS.

Procédure

1. Téléchargez les travaux personnalisés. Dans l'onglet **Définitions de personnalisation**, sélectionnez les travaux à télécharger et cliquez sur **Exécuter**.
2. Téléchargez les travaux vers le serveur FTP sur votre système z/OS. Spécifiez les informations obligatoires dans le panneau **Télécharger la définition de personnalisation**.
3. Cliquez sur **Terminer**.
4. Exécutez les travaux personnalisés. Cliquez sur l'onglet **Instructions de personnalisation** et suivez les consignes correspondant à chaque travail.

Chapitre 5. Planifier le déploiement des applications

Avant de déployer des applications sur WebSphere eXtreme Scale, vous devez passer en revue les conditions matérielles et logicielles requises, les paramètres de réseau et d'optimisation, les configurations de déploiement, etc. Servez-vous de la liste de contrôle pour vérifier que votre environnement est prêt pour le déploiement d'applications.

Vous trouverez une discussion des pratiques recommandées pour la conception d'applications WebSphere eXtreme Scale dans l'article suivant de developerWorks : Principles and best practices for building high performing and highly resilient WebSphere eXtreme Scale applications.

Déploiement d'applications

Avant de commencer à utiliser WebSphere eXtreme Scale dans un environnement de production, les points suivants sont à prendre en considération afin d'optimiser le déploiement.

Planifier le déploiement des applications

Points à prendre en considération :

- le nombre de systèmes et de processeurs : combien faut-il dans l'environnement de machines physiques et de processeurs ?
- le nombre de serveurs : combien de serveurs eXtreme Scale pour héberger les mappes eXtreme Scale ?
- le nombre de partitions : la quantité de données stockées dans les mappes est l'un des facteurs déterminant le nombre de partitions nécessaires
- le nombre de fragments répliques : combien de fragments répliques sont requis pour chacun des fragments primaires du domaine ?
- réplication synchrone ou asynchrone : les données sont-elles si vitales qu'une réplication synchrone soit indispensable ? Ou bien, est-ce que les performances sont une priorité plus importante ? Dans ce cas, la réplication asynchrone s'impose
- la taille des segments : combien de données seront-elles stockées sur chaque serveur ?

Configuration matérielle et logicielle requise

Vue d'ensemble des conditions requises en termes de matériels et de systèmes d'exploitation. Bien que vous ne soyez pas tenu d'utiliser un niveau spécifique de matériel ou de système d'exploitation pour WebSphere eXtreme Scale, nous n'en fournissons pas moins sur le site de support du produit (page Configuration requise) une liste détaillée des matériels et logiciels officiellement pris en charge, classée par système d'exploitation. En cas de conflit entre les informations présentées par le Centre de documentation et celles figurant sur cette page, les informations fournies par le site Web sont prioritaires. Les conditions préalables répertoriées par le Centre de documentation sont fournies à titre informatif uniquement.

Voir la page Configuration requise.

Configuration matérielle

WebSphere eXtreme Scale ne requiert pas la présence d'un niveau spécifique de matériel. La configuration matérielle requise dépend du matériel pris en charge pour l'installation de Java Platform, Standard Edition que vous utilisez pour exécuter WebSphere eXtreme Scale. Si vous utilisez eXtreme Scale avec WebSphere Application Server ou une autre implémentation Java Platform, Enterprise Edition, la configuration matérielle requise par ces plateformes est suffisante pour WebSphere eXtreme Scale.

Configuration requise en matière de système d'exploitation

eXtreme Scale ne requiert pas la présence d'un système d'exploitation d'un niveau donné. Chaque implémentation Java SE et Java EE requiert un niveau différent du système d'exploitation ou des correctifs pour les problèmes identifiés lors du test de l'implémentation Java. Les niveaux nécessaires à ces implémentations sont suffisants pour eXtreme Scale.

Configuration requise pour WebSphere Application Server

Les clients et les serveurs eXtreme Scale qui s'exécutent dans un environnement réparti et les ObjectGrids locaux en mémoire sont pris en charge par WebSphere Application Server version 6.0.2 et les versions ultérieures.

Remarque : Pour pouvoir utiliser le fournisseur de cache dynamique, votre système doit respecter l'une des conditions minimales requises suivantes :

- WebSphere Application Server version 6.1.0.25 ou supérieure et correctif provisoire PK85622
- WebSphere Application Server version 7.0.0.3 ou supérieure et correctif provisoire PK85622

Pour plus d'informations, consultez la section Recommended fixes for WebSphere Application Server.

Autres conditions requises par le serveur d'applications

Les autres implémentations Java EE peuvent utiliser la phase d'exécution d'eXtreme Scale en tant qu'instance locale ou client pour les serveurs eXtreme Scale. Pour implémenter Java SE, vous devez utiliser la version 1.4.2 ou une version ultérieure.

Java Platform, Enterprise Edition : points à prendre en considération

Lors de la préparation de l'intégration de WebSphere eXtreme Scale en environnement Java Platform, Enterprise Edition, un certain nombre de points sont à prendre en considération : options de configuration, conditions requises et limitations, déploiement et gestion des applications.

Exécuter des applications eXtreme Scale en environnement Java EE

Une application Java EE peut se connecter à une application eXtreme Scale distante. En outre, l'environnement WebSphere Application Server permet le démarrage d'un serveur eXtreme Scale lors du démarrage d'une application sur le serveur d'applications.

Si vous utilisez un fichier XML pour créer une instance ObjectGrid et que ce fichier XML se trouve dans le module du fichier EAR, accédez à ce fichier à l'aide de la méthode getClass().getClassLoader().getResource("META-INF/objGrid.xml") afin d'obtenir un objet URL permettant de créer une instance ObjectGrid. Dans l'appel à la méthode, remplacez le nom du fichier XML utilisé.

Vous pouvez utiliser des beans de démarrage pour que, à son démarrage, une application amorce une instance ObjectGrid et supprime cette instance lorsqu'elle s'arrête. Un bean de démarrage est un bean session sans état avec un emplacement distant com.ibm.websphere.startupservice.AppStartUpHome et une interface distante com.ibm.websphere.startupservice.AppStartUp. L'interface distante possède deux méthodes : la méthode start et la méthode stop. La méthode start permet d'amorcer l'instance et la méthode stop de la détruire. L'application utilise la méthode ObjectGridManager.getObjectGrid pour maintenir la référence à cette instance. Pour plus d'informations, voir dans le *Guide de programmation* les explications sur la manière d'accéder à un ObjectGrid avec ObjectGridManager.

Utiliser des loaders de classes

Lorsque les modules d'application qui utilisent des loaders de classes différents partagent une même instance ObjectGrid dans une application Java EE, vérifiez que les objets qui sont stockés dans eXtreme Scale et que les plug-in du produit se trouvent bien dans un Loader commun au sein de l'application.

Gérer dans un servlet le cycle de vie des instances ObjectGrid

Pour gérer dans un servlet le cycle de vie d'une instance ObjectGrid, vous pouvez utiliser la méthode init pour créer l'instance et la méthode destroy pour la supprimer. Si l'instance est mise en cache, elle est extraite et manipulée dans le code du servlet. Pour plus d'informations, voir dans le *Guide de programmation* les explications sur la manière d'accéder à un ObjectGrid avec ObjectGridManager.

Topologie des mises en cache : mise en cache en mémoire et mise en cache par répartition

Avec WebSphere eXtreme Scale, l'architecture de votre système peut utiliser la mise en cache des données locales en mémoire ou la mise en cache des données client-serveur réparties.

WebSphere eXtreme Scale requiert une infrastructure supplémentaire minimale pour pouvoir fonctionner. Cette infrastructure consiste en des scripts permettant d'installer, de démarrer et d'arrêter une application Java Platform, Enterprise Edition sur un serveur. Les données mises en cache sont stockées dans le serveur eXtreme Scale et les clients se connectent à distance à ce serveur.

Les caches répartis permettent d'améliorer les performances, la disponibilité et l'évolutivité du système. Les topologies dynamiques utilisées pour les configurer permettent d'équilibrer automatiquement les serveurs. Vous pouvez également ajouter d'autres serveurs sans redémarrer les serveurs eXtreme Scale existants. Il est possible de créer des déploiements simples ou des déploiements plus vastes se chiffrant en téraoctets et comptant plusieurs milliers de serveurs.

Cache interne locale

Dans le cas le plus simple, eXtreme Scale peut être utilisé comme cache de grille de données locale (non répartie) en mémoire. Cette mise en cache locale peut s'avérer

particulièrement utile pour les applications au nombre d'accès simultanés élevé où plusieurs unités d'exécution doivent accéder aux données temporaires et les modifier. Les données conservées dans une grille locale eXtreme Scale peuvent être indexées et extraites à l'aide du support de requête de WebSphere eXtreme Scale. La fonction d'interrogation des données peut représenter un outil précieux pour les développeurs utilisant des fichiers volumineux stockés dans la mémoire contrairement au support de structure de données limité offert par la machine virtuelle Java (JVM), prête à l'utilisation en l'état.

La topologie de cache local en mémoire de eXtreme Scale permet d'octroyer un accès cohérent et transactionnel aux données temporaires dans une machine virtuelle Java unique.

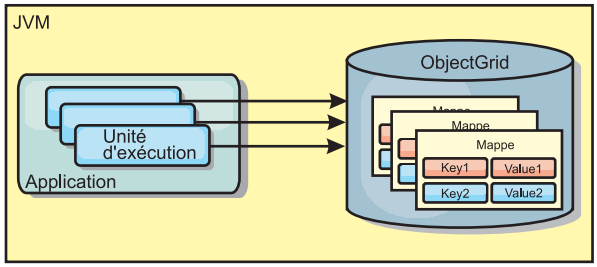


Figure 1. Scénario de cache local en mémoire

Avantages

- Configuration simple : une ObjectGrid peut être créée à l'aide d'un programme ou de manière déclarative avec le fichier XML du descripteur de déploiement ObjectGrid ou à l'aide d'une autre structure telle que Spring.
- Rapide : chaque mappe de sauvegarde peut être ajustée de façon indépendante pour optimiser l'utilisation de la mémoire et des accès simultanés.
- Configuration idéale pour les topologies de machine virtuelle Java dotées de petits jeux de données ou pour la mise en cache de données fréquemment consultées.
- Transactionnelle. Les mises à jour de mappe de sauvegarde peuvent être regroupées dans la même unité d'oeuvre et peuvent être intégrées en dernier lieu aux transactions constituées de deux phases telles que les transactions JTA (Java Transaction Architecture).

Inconvénients

- Aucune tolérance de panne.
- Les données ne sont pas répliquées. Les mémoires cache internes se prêtent aux données de référence en lecture seule.
- Non évolutive. La quantité de mémoire requise par la base de données peut dépasser la capacité de la machine virtuelle Java.
- Problèmes survenant lors de l'ajout de machines virtuelles Java :
  - Les données ne peuvent pas être facilement partitionnées ;
  - Nécessité de répliquer manuellement l'état entre les machines virtuelles Java ou chaque instance de cache peut présenter différentes versions des mêmes données.
  - L'invalidation est coûteuse.

– Chaque cache doit être préchauffé indépendamment. Le préchauffage est la période de chargement d'un jeu de données permettant de remplir le cache avec des données valides.

Utilisation

La topologie de déploiement de la mémoire cache interne locale ne doit être utilisée que lorsque la quantité de données à mettre en cache est limitée (peut être abritée par une seule machine virtuelle Java) et est relativement stable. Cette approche doit tolérer les données obsolètes. L'utilisation d'expulseurs pour conserver les données les plus fréquemment ou récemment utilisées dans le cache peut contribuer à réduire la taille du cache et à accroître la pertinence des données.

Cache interne en local répliqué sur des homologues

Dans le cas d'un cache WebSphere eXtreme Scale local, vous devez vous assurer que le cache est bien synchronisé s'il existe plusieurs processus avec des instances indépendantes de cache. Pour ce faire, activez avec JMS un cache répliqué sur des homologues.

WebSphere eXtreme Scale comprend deux plug-in qui propagent automatiquement les modifications de transactions entre les instances ObjectGrid homologues. Le plug-in JMSObjectGridEventListener propage automatiquement les modifications eXtreme Scale à l'aide de Java Messaging Service (JMS).

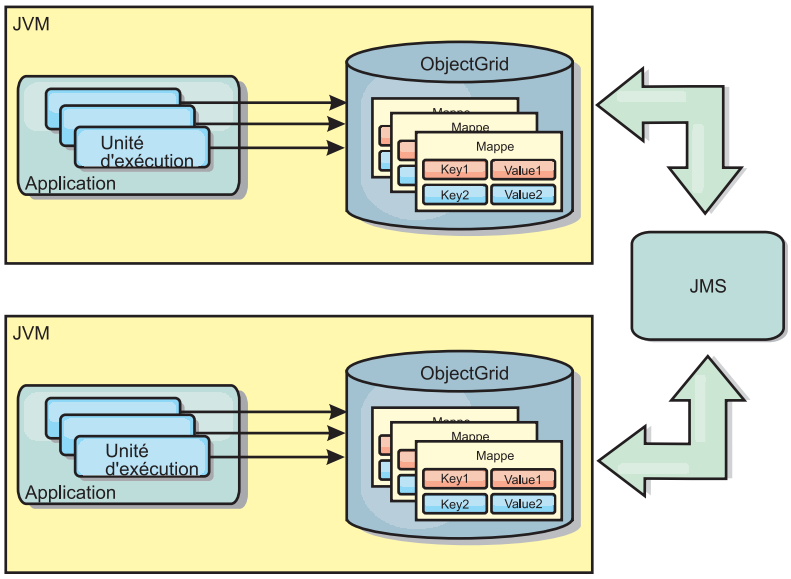


Figure 2. Cache répliqué sur des homologues avec des modifications qui sont propagées à l'aide de JMS

Si vous tournez en environnement WebSphere Application Server, vous pouvez également utiliser le plug-in TranPropListener. Ce plug-in utilise le gestionnaire de haute disponibilité (HA) pour propager les modifications à chacune des instances de cache eXtreme Scale homologue.

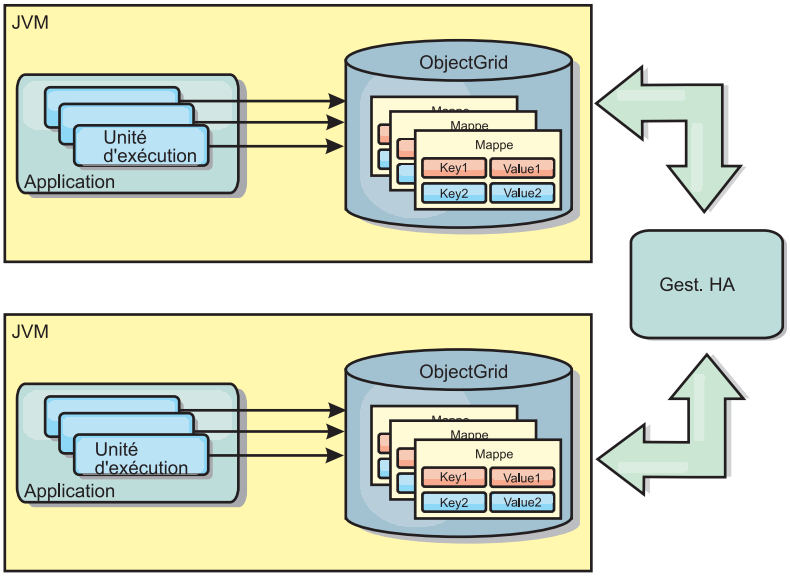


Figure 3. Cache répliqué sur des homologues avec des modifications qui sont propagées à l'aide du gestionnaire de haute disponibilité

Avantages

- Plus grande validité des données car celles-ci sont actualisées plus souvent.
- Avec le plug-in TranPropListener, tout comme avec l'environnement local, il est possible de créer la grille de données eXtreme Scale par programmation ou de manière déclarative avec le fichier XML du descripteur de déploiement d'eXtreme Scale ou avec d'autres structures de travail comme Spring. L'intégration au gestionnaire de haute disponibilité s'effectue automatiquement.
- Chaque mappe de sauvegarde peut être optimisée indépendamment en termes d'utilisation de la mémoire et de simultanéité des accès.
- Il est possible de regrouper en une seule unité d'oeuvre les mises à jour des mappes de sauvegarde qui peuvent être intégrées comme derniers participants de transactions en deux phases comme le sont les transactions Java Transaction Architecture (JTA).
- Idéal pour les topologies comprenant un nombre restreint de machines virtuelles Java avec un dataset de taille raisonnablement réduite ou pour la mise en cache des données à accès fréquent.
- Les modifications de la grille de données eXtreme Scale sont répliquées à toutes les instances eXtreme Scale homologues. Les modifications sont cohérentes tant qu'un abonnement durable est utilisé.

Inconvénients

- La configuration et la maintenance du plug-in JMSObjectGridEventListener peut s'avérer une tâche complexe. Il est possible de créer la grille de données eXtreme Scale par programmation ou de manière déclarative avec le fichier XML du descripteur de déploiement d'eXtreme Scale ou avec d'autres structures de travail comme Spring.
- Pas d'extensibilité : la quantité de mémoire requise par la base de données risque de submerger la machine virtuelle Java.
- Fonctionne de manière incorrecte lorsqu'on ajoute des machines virtuelles Java :
  - les données ne sont pas facilement partitionnées
  - l'invalidation est onéreuse

– chaque cache doit être prérempli de manière indépendante

Quand l'utiliser

Cette topologie de déploiement ne doit être utilisée que lorsque la quantité de données à mettre en cache est de taille réduite (qu'elle peut tenir sur une seule machine virtuelle Java) et qu'elles sont relativement stables.

Cache réparti

La plupart du temps, WebSphere eXtreme Scale est utilisé en tant que cache partagé permettant un accès transactionnel aux données de plusieurs composants là où une base de données classique aurait été nécessaire. Avec le cache partagé, il n'est plus nécessaire de configurer une base de données.

Le cache est cohérent car tous les clients y voient les mêmes données. Chaque donnée est stockée dans le cache sur un seul serveur ce qui permet d'éviter la coexistence de plusieurs copies d'enregistrements risquant de contenir des versions différentes des données. Un cache cohérent contient également davantage de données au fur et à mesure que des serveurs sont ajoutés à la grille et que le cache évolue de façon linéaire en même temps que la grille s'agrandit. Comme les clients accèdent aux données de cette grille à l'aide d'appels procéduraux, cette mémoire est également appelée cache distant (ou éloigné). Grâce au partitionnement des données, chaque processus contient un sous-ensemble unique de données. Les grilles de grande taille peuvent contenir davantage de données et gérer un plus grand nombre de demandes. Par ailleurs, il n'est plus nécessaire d'insérer les données d'invalidation autour de la grille car aucune donnée périmée n'existe. Le cache cohérent contient uniquement la copie la plus récente de chaque donnée.

Si vous exécutez un environnement WebSphere Application Server, le plug-in TranPropListener est aussi disponible. Il utilise le composant de haute disponibilité (gestionnaire HA) de WebSphere Application Server pour propager les modifications à chaque instance de cache ObjectGrid homologue.

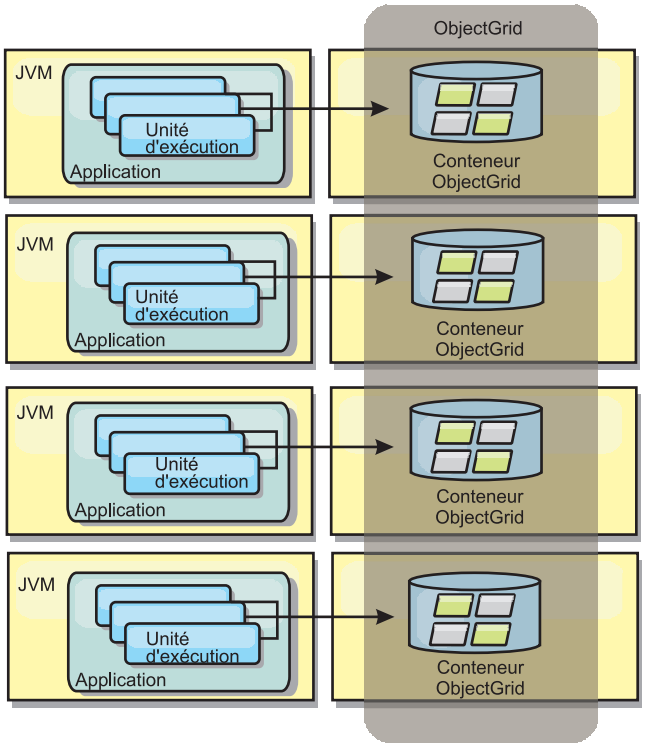


Figure 4. Cache réparti

Cache local

Lorsqu'eXtreme Scale est utilisé dans le cadre d'une topologie répartie, les clients peuvent éventuellement disposer d'un cache local en ligne. L'on appelle cache local ce cache facultatif. Il s'agit d'un ObjectGrid indépendant, présent sur chaque client et faisant office de cache du cache distant côté serveur. Il est activé par défaut lorsque le verrouillage est configuré sur OPTIMISTIC ou sur NONE. Son utilisation est impossible lorsque le verrouillage est configuré sur PESSIMISTIC.

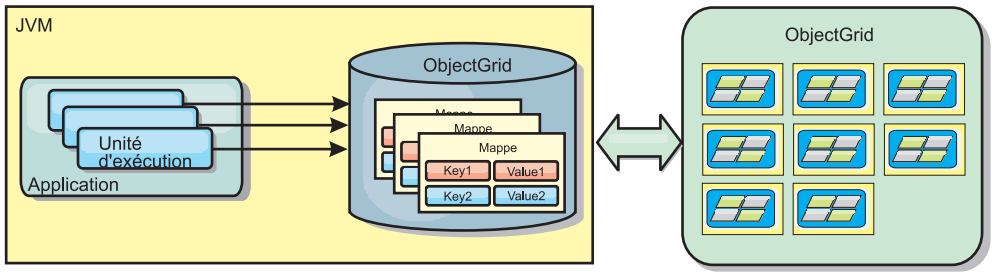


Figure 5. Cache local

Le cache local est très rapide car il offre un accès en mémoire à un sous-ensemble des données stockées à distance sur les serveurs eXtreme Scale. Il n'est pas partitionné et contient des données provenant de n'importe quelle partition eXtreme Scale distante. Jusqu'à trois groupes de caches peuvent exister dans WebSphere eXtreme Scale :

1. Le cache du groupe des transactions contient toutes les modifications apportées à une même transaction. Il contient une copie de travail des données jusqu'à ce que la transaction soit validée. Lorsqu'une transaction client demande des données à une ObjectMap, la transaction est vérifiée en priorité.

2. Le cache local du groupe des clients contient un sous-ensemble des données du groupe des serveurs. Lorsque le groupe des transactions ne contient pas de données, celles-ci, si elles existent, sont extraites du cache local et insérées dans le cache de transactions.
3. La grille du groupe des serveurs contient la majorité des données. Elle est partagée entre tous les clients. Le groupe des serveurs peut être partitionné, ce qui permet la mise en cache d'un grand nombre de données. Lorsque le cache local ne contient pas de données, celles-ci sont extraites du groupe des serveurs et insérées dans le cache du client. Le groupe des serveurs peut aussi avoir un plug-in Loader. Lorsque la grille ne contient pas les données demandées, le chargeur est appelé et les données résultantes sont insérées dans la grille à partir du magasin de données dorsal.

Pour désactiver le cache local, donnez la valeur 0 à l'attribut numberOfBuckets dans la configuration du descripteur eXtreme Scale des remplacements par le client. Pour plus d'informations sur les stratégies de verrouillage dans eXtreme Scale, consultez la rubrique relative au verrouillage des entrées de mappe. Le cache local peut également être configuré de façon à utiliser d'autres règles d'expulsion et des plug-in différents qui utilisent une configuration de descripteur eXtreme Scale des remplacements par les clients.

Avantage

- rapidité du temps de réponse, car tous les accès aux données se font localement

Inconvénients

- longévité plus importante des données périmées
- obligation d'utiliser un expulseur pour invalider les données et éviter ainsi de manquer de mémoire

Utilisation

A utiliser lorsque le temps de réponse est élevé et que la présence de données périmées est tolérée.

Cache imbriqué

Les grilles eXtreme Scale peuvent s'exécuter dans des processus existants tels que des serveurs eXtreme Scale imbriqués. Elles peuvent également être gérées en tant que processus externes. Les grilles imbriquées sont utiles lorsque l'exécution se fait dans un serveur d'applications tel que WebSphere Application Server. Vous pouvez démarrer les serveurs eXtreme Scale non imbriqués à l'aide de scripts de ligne de commande et les exécuter dans un processus Java.

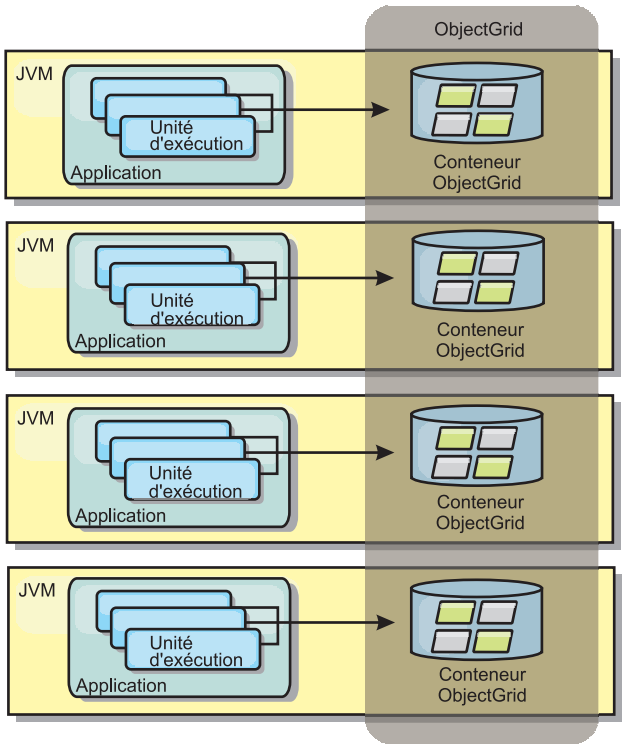


Figure 6. Cache imbriqué

Avantages

- simplification de l'administration en raison du nombre inférieur de processus à gérer
- simplification du déploiement d'applications en raison de l'utilisation par la grille du chargeur de classe d'application du client
- prise en charge du partitionnement et de la haute disponibilité

Inconvénients

- augmentation de l'encombrement mémoire dans le processus client car toutes les données sont regroupées dans le processus
- augmentation de l'utilisation de l'unité centrale en vue de la gestion des demandes des clients
- plus grande difficulté à gérer les mises à niveau des applications car les clients utilisent les mêmes fichiers d'archive Java que les serveurs
- moindre flexibilité. Les clients et les serveurs de grille ne peuvent évoluer au même rythme. Lorsque des serveurs sont définis en externe, la gestion du nombre de processus devient plus flexible

Utilisation

Utilisez les grilles imbriquées lorsqu'une grande quantité de mémoire est disponible dans le processus client pour les données de la grille et pour les données de basculement.

Pour plus d'informations, consultez la rubrique relative à l'activation du mécanisme d'invalidation du client dans le manuel *Guide d'administration*.

Topologies de réplication de grilles multi-maîtres (PA)

La réplication asynchrone multi-maître permet à deux grilles, voire plus, de devenir des miroirs exacts les unes des autres. Cette mise en miroir est effectuée à l'aide d'une réplication asynchrone entre des liens interconnectant les grilles. Chaque grille est hébergée au sein d'un domaine complètement indépendant, possédant son propre service de catalogue et ses propres serveurs conteneurs et portant un nom exclusif. La réplication asynchrone multi-maître permet d'utiliser des liens pour interconnecter une collection de ces domaines et de synchroniser ensuite ces derniers grâce, précisément, à la réplication via ces liens. eXtreme Scale permet de construire quasiment n'importe quelle topologie, car la définition des liens entre les domaines est laissée à l'appréciation et à l'initiative des administrateurs.

7.1+ La réplication de grilles multi-maîtres est une importante nouveauté de la version 7.1.

Les domaines : des grilles avec des caractéristiques spécifiques

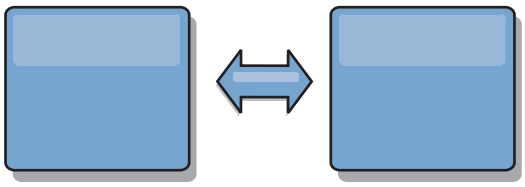
Les grilles utilisées dans les topologies de réplication multi-maître sont également désignées sous le nom de *domaines*. Chaque domaine doit avoir les caractéristiques suivantes :

- disposer d'un service de catalogue privé avec un nom de domaine exclusif
- avoir le même nom de grille que les autres grilles du domaine
- avoir le même nombre de partitions que les autres grilles du domaine
- être une grille FIXED_PARTITION (les grilles PER_CONTAINER ne peuvent être répliquées)
- avoir le même nombre de partitions (sans forcément pour autant avoir le même nombre et le même type de fragments répliques)
- avoir les mêmes types de données à répliquer que les autres grilles du domaine
- avoir le même nom de groupes de mappes (mapsets), le même nom de mappes et les mêmes modèles de mappes dynamiques que les autres grilles du domaine

Tous les groupes de mappes ayant les caractéristiques énoncées ci-dessus seront répliqués après que les domaines de services de catalogue auront été démarrés. Voir «Topologies de réplication de grilles multi-maîtres (PA)» pour connaître les groupes de mappes qui ne seront pas répliquées.

Liens connectant des domaines

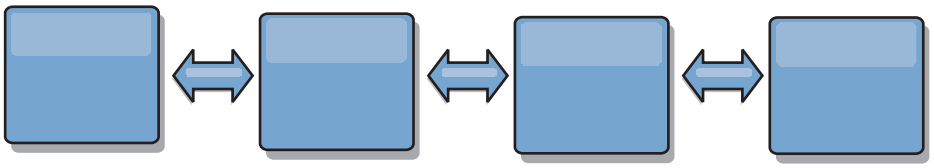
Une infrastructure de grilles de réplication est un graphe connecté de domaines reliés par des liens bidirectionnels. Un lien permet à deux domaines d'échanger des modifications de données. Ainsi, la topologie la plus simple sera une paire de domaines reliés par un seul lien. Les domaines sont nommés en partant de A, puis B, et ainsi de suite, à partir de la gauche. Le lien peut traverser un réseau WAN étendu, couvrant de longues distances. En cas d'interruption du lien, les modifications peuvent toujours être apportées aux données dans l'un ou l'autre des deux domaines. La réconciliation des modifications s'effectue plus tard, lorsque le lien recommence à connecter les domaines. Les liens tentent automatiquement de se reconnecter si la connexion réseau est interrompue.



Une fois que les liens ont été définis, eXtreme Scale va tout d'abord tenter de rendre identique chaque domaine, puis il va essayer ensuite de les maintenir dans un état identique lorsque des modifications se produiront dans l'un de ces domaines. L'objectif d'eXtreme Scale est que chaque domaine soit un miroir exact de tout autre domaine connecté par les liens. Les liens de réplication entre les domaines aident à garantir que toute modification effectuée dans un domaine est copiée vers les autres domaines.

Topologies linéaires

Bien qu'elle figure au rang des topologies les plus simples, la topologie linéaire illustre un certain nombre de qualités des liens. Tout d'abord, il n'est pas nécessaire qu'un domaine soit directement connecté à chacun des autres domaines pour recevoir des modifications. Le domaine B ira prendre des modifications dans le domaine A. Le domaine C reçoit les modifications du domaine A via le domaine B, lequel connecte les domaines A et C. De la même manière, le domaine D reçoit les modifications des autres domaines via le domaine C. De ce fait, la charge de la répartition des modifications est distribuée et elle n'incombe plus à la seule source de ces modifications.



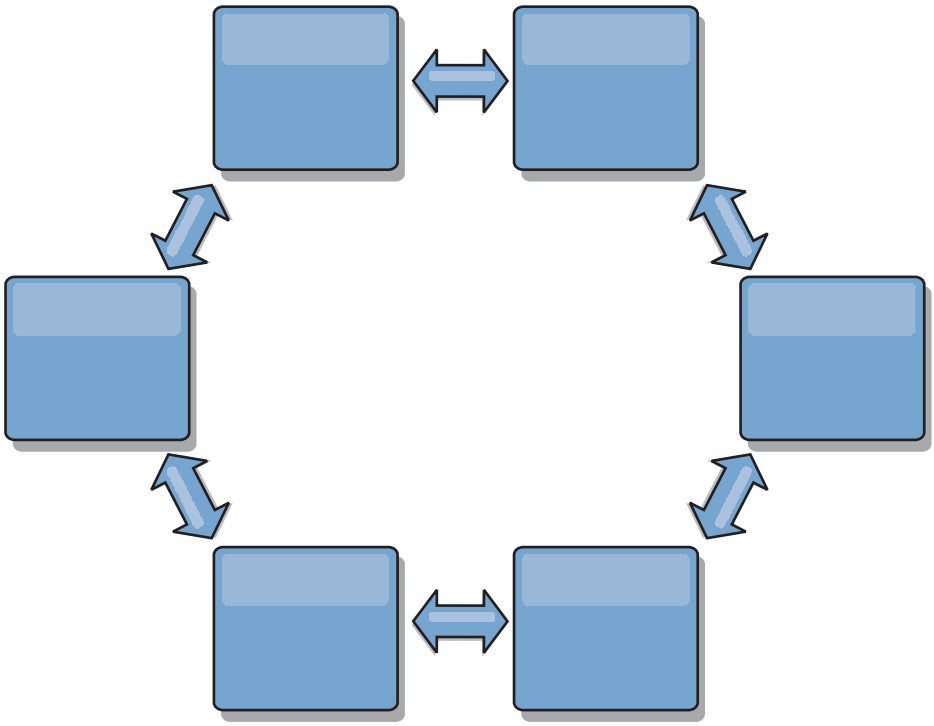
Et si le domaine C tombait en panne, les événements suivants se produiraient :

1. Le domaine D serait orphelin jusqu'au redémarrage du domaine C.
2. Le domaine C se synchroniserait avec le domaine B, lequel est une copie du domaine A.
3. Le domaine D utiliserait le domaine C pour se synchroniser avec les modifications intervenues sur les domaines A et B pendant que le domaine D était orphelin (et que le domaine C était arrêté).

A la fin, les domaines A, B, C et D redeviendraient tous identiques.

Topologies en anneau

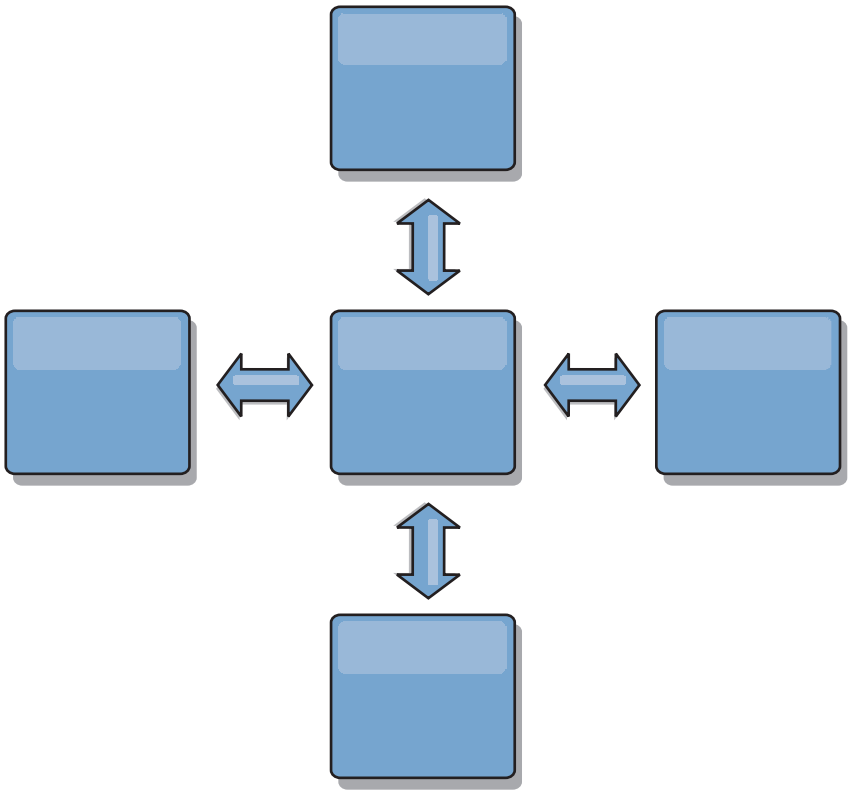
Les topologies en anneau sont un exemple de topologie encore plus résilientes. En effet, la défaillance d'un domaine ou d'un lien n'empêche pas les domaines survivants de continuer à obtenir des modifications en circulant autour de l'anneau et en s'éloignant de la défaillance. Chaque domaine a deux liens vers les autres domaines. Chaque domaine a au maximum deux liens, quelle que soit la taille de la topologie en anneau. Le délai de propagation des modifications peut être important, car les modifications d'un domaine particulier peuvent avoir besoin de traverser plusieurs domaines avant que la totalité des domaines ne puisse voir la totalité des modifications. Une topologie linéaire a le même problème.



Représentez-vous une topologie en anneau plus sophistiquée, avec un domaine racine au centre de l'anneau. Le domaine racine joue le rôle de chambre centrale de compensation tandis que les autres domaines font office de chambres distantes de compensation pour les modifications se produisant dans le domaine racine. Le domaine racine peut arbitrer les modifications entre les domaines. Si une topologie en anneau contient plusieurs anneaux autour d'un domaine racine, ce dernier ne pourra arbitrer les modifications que dans les domaines de l'anneau le plus proche du centre. Mais les résultats de l'arbitrage seront ventilés vers les domaines des autres anneaux.

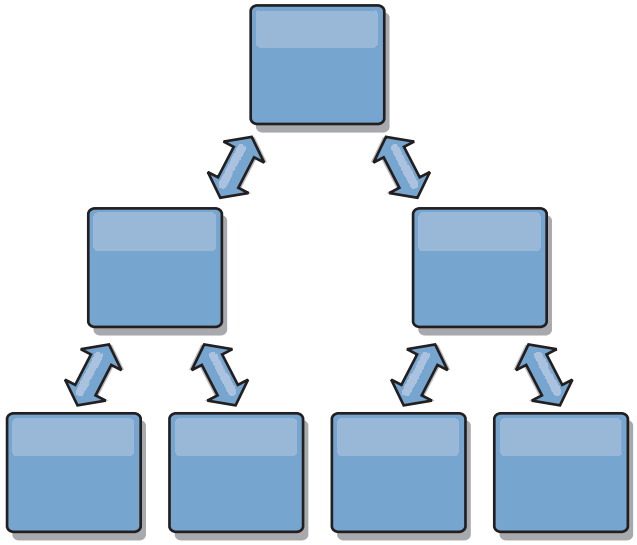
Topologies en étoile

Si, dans une topologie en étoile, les délais d'attente sont moindres, les modifications voyageant au maximum sur un domaine intermédiaire (le concentrateur), cette topologie ne va pas sans autres problèmes. Elle a un domaine central qui fait office de « moyeu » ou de concentrateur. Ce domaine concentrateur est connecté via un lien à chacun des domaines qui jouent le rôle de « rayons » de la roue. On le voit, la charge de la répartition des modifications entre les domaines repose toute entière sur le concentrateur. Ce dernier fait office de chambre de compensation en cas de collisions, ce qui peut s'avérer important dans certains scénarios. Dans un environnement soumis à une fréquence élevée de modifications, pour pouvoir rester actif et opérationnel, le concentrateur peut avoir besoin de s'exécuter sur plus de matériels que les autres noeuds. eXtreme Scale est conçu pour évoluer de manière linéaire, ce qui signifie que l'on peut, si nécessaire, étoffer le concentrateur sans difficultés. Mais, si le concentrateur vient à tomber en panne, il faudra attendre qu'il ait redémarré pour que les changements puissent être répartis. Toutes les modifications intervenues sur les domaines autres que le concentrateur seront réparties après la reconnexion de ce dernier.



Topologies en arbre

Dernier exemple de topologie : une arborescence acyclique dirigée. Par acyclique, l'on entend qu'aucun cycle (aucune boucle) ne se passe sur l'arborescence. Dirigée signifie qu'il n'existe de liens qu'entre des parents et des enfants. Cette configuration peut être utile pour les topologies comprenant tant de domaines qu'il ne serait pas pratique d'avoir un concentrateur connecté de manière centralisée à chaque ordinateur possible. Autre cas de figure où cette topologie trouve toute son utilité : le besoin d'ajouter des domaines enfants sans avoir à modifier le domaine racine.



Cette topologie peut toujours avoir une chambre de compensation centrale avec le domaine racine, mais le deuxième niveau peut faire office de chambres de compensation pour les modifications se produisant dans le domaine situé au niveau inférieur. Le domaine racine ne peut arbitrer que les modifications entre les domaines de ce deuxième niveau. Des arbres n-aires sont également possibles. Un arbre n-aire a n enfants à chaque niveau. Chaque domaine a un déploiement de n.

Points concernant l'arbitrage à prendre en considération dans la conception des topologies

Des collisions entre des modifications peuvent se produire s'il est possible à des enregistrements identiques d'être modifiés simultanément en deux endroits différents. Configurez chaque domaine pour qu'ils aient tous la même quantité de processeurs, de mémoire et de ressources réseau. Vous vous rendrez sans doute compte que les domaines gérant les collisions de modifications (arbitrage) utilisent plus de ressources que les autres domaines. Les collisions sont détectées de manière automatique. Deux mécanismes permettent de les résoudre :

- **Arbitre par défaut**. Le protocole par défaut est d'utiliser les modifications provenant du domaine dont le nom vient en premier par ordre alphabétique. Supposons par exemple que les domaines A et B génèrent un conflit pour un enregistrement, dans ce cas, la modification du domaine B sera ignorée. Le domaine A conserve sa version et l'enregistrement dans le domaine B est modifié de manière à correspondre à celui du domaine A. Cela s'applique également pour les applications où les utilisateurs ou les sessions sont liées de manière normale ou ont une affinité avec l'une des grilles.
- **Arbitre personnalisé**. Les applications peuvent très bien fournir un arbitre personnalisé. Lorsqu'un domaine détecte une collision, il fait appel à l'arbitre. Pour savoir comment développer un arbitre personnalisé de bonne qualité, voir Développement d'arbitres personnalisés pour la réplication multi-maître.

Si des collisions doivent être possibles dans les topologies que vous envisagez, pensez à utiliser une topologie en étoile ou en arbre. Il est en effet plus facile dans ces deux topologies d'éviter des collisions sans fin, ce qui peut arriver lorsque :

1. plusieurs domaines subissent une collision
2. chaque domaine résout la collision en local, ce qui produit des révisions
3. les révisions entrent en collision, d'où des révisions de révisions
4. et ainsi de suite, au fur et à mesure que les révisions sont propagées dans les divers domaines tout en essayant d'atteindre la synchronicité

Pour éviter des collisions sans fin, choisissez un domaine spécifique – un *domaine d'arbitrage* – comme gestionnaire des collisions d'un sous-ensemble de domaines. Exemple : une topologie en étoile pourra utiliser le concentrateur comme gestionnaire des collisions. Le gestionnaire de collisions ignore les collisions détectées par les sous-domaines. Le domaine du concentrateur va créer des révisions en empêchant les révisions de collisions qui échappent à tout contrôle. Le domaine qui s'est vu attribuer la gestion des collisions doit se lier à tous les domaines dont il est chargé de résoudre les collisions. Dans une topologie en arbre, tous les domaines parents internes résolvent les collisions pour leurs enfants immédiats. Par contraste, si vous utilisez une topologie en anneau, vous ne pouvez désigner un domaine de l'anneau comme résolvant les collisions.

Le tableau qui suit récapitule les approches en matière d'arbitrage qui sont les plus compatibles avec les diverses topologies.

Tableau 5. Approches en matière d'arbitrage. Ce tableau énonce si l'arbitrage entre applications est compatible avec les diverses topologies.

Topologie	Arbitrage entre applications ?	Commentaires
Linéaire à deux domaines	Oui	Choisissez l'un des domaines comme arbitre.
Linéaire à trois domaines	Oui	L'arbitre doit être le domaine du milieu. Pensez à ce domaine comme au concentrateur d'une topologie en étoile simple.
Linéaire à plus de trois domaines	Non	L'arbitrage entre applications n'est pas pris en charge.
Concentrateur avec n "rayons"	Oui	Le domaine d'arbitrage doit être le concentrateur avec des liens vers tous les "rayons".
Anneau de n domaines	Non	L'arbitrage entre applications n'est pas pris en charge.
Arbre dirigé acyclique (arbre n-aire)	Oui	Tous les noeuds racine ne doivent arbitrer que leurs descendants directs.

Points concernant les liens à prendre en considération dans la conception des topologies

Dans l'idéal, une topologie comprend le minimum de liens tout en optimisant les compromis entre les temps d'attente des modifications, la tolérance aux pannes et les caractéristiques de performances.

- **Temps d'attente des modifications**

  Les temps d'attente des modifications sont déterminés par le nombre de domaines intermédiaires que doivent traverser les modifications avant d'arriver à destination.

  Les topologies qui offrent les meilleurs temps d'attente sont celles qui éliminent les domaines intermédiaires en liant chacun des domaines à chacun des autres domaines. Mais un domaine doit effectuer la réplication à proportion du nombre de ses liens. Pour les topologies de grande taille, le nombre de liens à définir peut constituer une lourde charge pour les administrateurs.

  La vitesse à laquelle est une modification copiée vers les autres domaines dépend de facteurs supplémentaires :

  – le processeur et la bande passante réseau sur le domaine source
  – le nombre de domaines intermédiaires et de liens entre le domaine source et le domaine cible
  – les ressources de processeur et de réseau utilisables par les domaines source, cible et intermédiaires

- **Tolérance aux pannes**

  La tolérance aux pannes est déterminée par le nombre de chemins existant entre deux domaines pour la réplication des modifications.

  S'il n'existe qu'un seul lien entre les domaines, en cas de défaillance du lien, les modifications ne seront pas propagées. Si ce seul lien entre deux domaines passe par des domaines intermédiaires, les modifications ne seront pas non plus propagées si l'un des domaines intermédiaires tombe en panne.

  Prenons la topologie linéaire à trois domaines, A, B et C :

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

Si l'une des situations suivantes se présente, le domaine C ne verra pas les modifications de A :

– le domaine A est actif et le domaine B est arrêté
– le lien entre A et B ne fonctionne pas
– le lien entre B et C ne fonctionne pas

Par contraste, une topologie en anneau permet à chaque domaine d'extraire les modifications dans un sens ou dans l'autre.

```
A <-> B <-> C <-> retour vers A
```

Par exemple, si le domaine B est arrêté, le domaine C continue d'extraire les modifications directement depuis le domaine A.

Une conception en étoile dépend du concentrateur par qui passent toutes les modifications ; s'il s'arrête, tout s'arrête. Mais il ne faut pas oublier qu'un domaine reste une grille totalement tolérante aux pannes et qui peut souffrir des défaillances du réseau WAN ou des centres de données physiques.

- **Performances**

  Le nombre des liens définis sur un domaine affecte les performances. Plus de liens utilisent davantage de ressources, et cela peut se traduire par une chute des performances de la réplication. La capacité d'un domaine A à extraire les modifications via d'autres domaines décharge efficacement ce domaine A de répliquer partout ses transactions. *La charge de la répartition des modifications sur un domaine est limitée au nombre des liens qu'utilise ce domaine. Cela n'a rien à voir avec le nombre de domaines dans la topologie.* Cette propriété permet l'évolutivité et autorise à partager la charge de la répartition des modifications entre les domaines de la topologie, plutôt que d'imposer cette charge à un seul domaine.

  Un domaine peut extraire les modifications indirectement via d'autres domaines. Prenons une topologie linéaire à cinq domaines :

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

  – A extrait les modifications de B, C, D, et E via B
  – B extrait les modifications directement de A et de C et les modifications de D et de E via C
  – C extrait les modifications directement de B et de D et les modifications de A via B et de E via D
  – D extrait les modifications directement de C et de E et les modifications de A et de B via C
  – E extrait les modifications directement de D et et les modifications de A, B et C via D

  La charge de la répartition sur les domaines A et E est la plus faible, car ces domaines ont chacun un seul lien avec un seul domaine. La charge de la répartition sur les domaines B, C et D est le double de celle sur les domaines A et E car B, C et D ont chacun un lien avec deux domaines. Cette répartition des charges demeurerait constante même si la topologie linéaire contenait 1 000 domaines, car la charge dépend du nombre de liens de chaque domaine et non du nombre globale de domaines dans la topologie.

Points à prendre en considération concernant les performances

Les limitations suivantes sont à prendre en compte pour les topologies de réplication multi-maître.

- **Optimisation de la répartition des modifications** (abordée plus haut).
- **Temps d'attente de la réplication** (abordée plus haut).

- **Performances des liens de réplication** eXtreme Scale crée un seul socket TCP/IP entre n'importe quelle paire de machines virtuelles Java. Tout le trafic entre ces machines virtuelles Java passe par ce socket, y compris la réplication multi-maître. Les domaines étant hébergés sur au moins n machines virtuelles Java conteneurs, fournissant au moins n liens TCP vers les domaines homologues, les domaines ayant le plus grand nombre de conteneurs sont ceux qui offrent les plus hauts niveaux de performances pour la réplication. Plus de conteneurs est synonyme de davantage de ressources processeur et réseau.
- **Optimisation de la fenêtre dynamique TCP et RFC 1323** Activer la prise en charge de la RFC 1323 aux deux extrémités d'un lien permet à davantage de données d'effectuer des aller-retour, ce qui donne des débits plus élevés. Cette technique étend les capacités de la fenêtre d'un facteur d'environ 16 000.

  N'oubliez pas que les sockets TCP utilisent un mécanisme de fenêtre dynamique pour contrôle le flux des données en vrac, qui limite normalement le socket à 64 Ko pour un intervalle d'aller-retour. Si cet intervalle est de 100 ms, la bande passante est limitée à 640 Ko/s sans optimisation supplémentaire. L'utilisation intégrale de la bande passante disponible sur un lien peut nécessiter une optimisation qui est spécifique au système d'exploitation. La plupart des systèmes d'exploitation comportent des paramètres d'optimisation, y compris des options RFC 1323, permettant d'améliorer les débit sur les liens à hauts temps d'attente.

  Plusieurs facteurs peuvent affecter les performances de la réplication :
  - la vitesse à laquelle eXtreme Scale peut extraire les modifications
  - la vitesse à laquelle eXtreme Scale peut extraire les demandes de réplication
  - la capacité de la fenêtre dynamique
  - l'optimisation de la mémoire tampon réseau des deux côtés d'un lien pour permettre à eXtreme Scale d'extraire les modifications sur le socket à la vitesse maximale possible
- **Sérialisation des objets** Toutes les données doivent être sérialisables. Si un domaine n'utilise pas COPY_TO_BYTES, il doit utiliser la sérialisation Java ou ObjectTransformers pour optimiser les performances de la sérialisation.
- **Compression** eXtreme Scale compresse par défaut toutes les données envoyées entre les domaines. La version actuelle ne permet pas de désactiver la compression.
- **Optimisation de la mémoire** *L'utilisation de la mémoire pour une topologie de réplication multi-maître est largement indépendante du nombre de domaines présents dans la topologie.*

  Activer la réplication multi-maître ajoute du temps système fixe par entrée de mappe pour gérer la vérification des versions. Chaque conteneur suit également une quantité fixe de données pour chacun des domaines de la topologie. Une topologie à deux domaines utilise approximativement la même mémoire qu'une topologie à cinquante domaines. eXtreme Scale n'utilise pas dans son implémentation de replay logs ou d'autres files d'attente du même genre. Cela veut dire que, si un lien de réplication est indisponible pendant une période assez longue, il n'y a aucune structure de données en train de croître en taille dans l'attente d'une reprise de la réplication au redémarrage du lien.

Centres de données multiples avec FIXED_PARTITION

Vous pouvez à présent utiliser une grille FIXED_PARTITION entre au moins deux centres de données. Chaque centre de données a besoin de son propre domaine, en termes de réplication multi-maître. Chaque centre de données peut lire et écrire des données sur le domaine local. Ces modifications seront propagées vers l'autre

centre de données à l'aide des liens que vous aurez définis.

Clients intégralement répliqués

Cette variante de la topologie implique une paire de serveurs eXtreme Scale s'exécutant comme concentrateur. Chaque client crée une grille à conteneur unique autonome avec un catalogue dans sa machine virtuelle Java. Un client utilise sa grille pour se connecter au catalogue du concentrateur, ce qui provoque la synchronisation du client avec le concentrateur dès que le client obtient une connexion au concentrateur.

Toutes les modifications effectuées par le client sont locales pour le client et elles sont répliquées vers le concentrateur de manière asynchrone. Le concentrateur joue le rôle de domaine d'arbitrage, répartissant les modifications à tous les clients connectés. La topologie de clients intégralement répliqués fournit un bon cache de niveau 2 pour un associateur relationnel d'objets comme OpenJPA. Les modifications seront réparties rapidement via le concentrateur entre les machines virtuelles Java clients. Tant que la taille du cache peut être contenue dans l'espace de segment mémoire disponible des clients, cette topologie est une architecture tout à fait indiquée pour ce style de cache de niveau 2.

Si nécessaire, utilisez plusieurs partitions pour échelonner le domaine concentrateur sur plusieurs machines virtuelles Java. Toutes les données devant tenir sur une seule machine virtuelle Java, l'utilisation de partitions multiples augmente la capacité du concentrateur à répartir et à arbitrer les modifications, mais elle ne change pas la capacité d'un domaine unique.

Limitations

Les limitations suivantes sont à prendre en compte pour décider s'il y a lieu d'utiliser des topologies de réplication multi-maître et, si oui, quand.

- **Précautions à prendre dans la configuration de chargeurs de classes avec plusieurs domaines**

  Les domaines doivent avoir accès à toutes les classes qui sont utilisées comme clés et comme valeurs. Toutes les dépendances doivent être reflétées dans tous les chemins de classes des machines virtuelles Java conteneurs de grille de la totalité des domaines. Si un plug-in CollisionArbiter extrait la valeur d'une entrée de cache, les classes correspondant aux valeurs doivent être présentes pour le domaine qui fait appel à l'arbitre.
- **L'utilisation de chargeurs n'est pas recommandée**

  Les chargeurs peuvent servir à l'interfaçage des modifications entre une grille et une base de données. Il y a fort peu de probabilités que toutes les grilles (domaines) d'une topologie cohabitent géographiquement avec la même base de données. De toute façon, vu les temps d'attente du WAN et d'autres facteurs, ce cas de figure est peu souhaitable.

  Autre problème qui nécessite que l'on aborde la conception avec précaution : le préchargement des grilles. D'ordinaire, lorsqu'elle redémarre, une grille est préchargée à nouveau. Le préchargement n'est pas nécessaire ni même souhaitable lorsqu'on utilise la réplication multi-maître. Dès qu'un domaine est en ligne, il se recharge automatiquement avec le contenu des domaines auxquels il est lié. Il en découle qu'il n'est pas nécessaire de lancer un préchargement manuel d'une grille qui est un domaine dans une topologie de réplication multi-maître.

Les chargeurs obéissent en général à des règles d'insertion et d'actualisation. Dans le cadre de la réplication multi-maître, les insertions doivent être traitées comme des fusions. Lorsque les données sont extraites à distance après le redémarrage d'un domaine, les données existantes seront "insérées" dans le domaine local. Comme il se peut que ces données se trouvent déjà dans la base de données locale, une insertion classique dans la base de données échouera avec une exception de clé en double. Plutôt que des insertions classiques, il faut utiliser une sémantique des fusions.

Il est possible de configurer eXtreme Scale pour qu'il effectue des préchargements en fonction des fragments en utilisant les méthodes preload des plug-in Loader. N'utilisez pas cette technique dans une topologie de réplication multi-maître. Utilisez plutôt un préchargement en fonction du client lorsque la topologie est démarrée (initialement). Autorisez la topologie multi-maître à actualiser les domaines redémarrés à l'aide d'une copie actuelle de ce qui est stocké dans les autres domaines de la topologie. Après que les domaines ont été redémarrés, la responsabilité de leur synchronisation incombera à la topologie multi-maître.

- **Pas de prise en charge d'EntityManager**

  Un groupe de mappes contenant une mappe d'entités n'est pas répliqué entre les domaines.
- **Pas de prise en charge des mappes de tableaux d'octets**

  Un groupe de mappes contenant une mappe qui est configurée avec COPY_TO_BYTES n'est pas répliqué entre les domaines.
- **Pas de prise en charge de l'écriture différée**

  Un groupe de mappes contenant une mappe qui est configurée avec la prise en charge de l'écriture différée n'est pas répliqué entre les domaines.

Configurations de déploiement pour eXtreme Scale

Vous pouvez déployer WebSphere eXtreme Scale dans deux types d'environnement : un environnement local ou un environnement réparti. La configuration nécessaire dépend du type de l'environnement de déploiement.

Environnement local

Lorsque vous procédez à un déploiement dans un environnement local, eXtreme Scale s'exécute dans un seule machine virtuelle Java et n'est pas répliqué. L'environnement local requiert un fichier XML ObjectGrid ou des API ObjectGrid.

Environnement réparti

Lorsque vous procédez à un déploiement dans un environnement réparti, eXtreme Scale s'exécute sur plusieurs machines virtuelles Java. Dans cette configuration, vous pouvez utiliser les fonctions de réplication et de partitionnement des données. Un environnement réparti peut aussi être considéré comme une topologie de déploiement dynamique dans laquelle vous pouvez ajouter des serveurs selon vos besoins. Comme dans le cas d'un environnement local, un fichier XML ObjectGrid ou une configuration par programmation équivalente est nécessaire dans un environnement réparti. Vous devez en outre fournir un fichier XML de règle de déploiement contenant les détails de la configuration de votre grille de données en mémoire eXtreme Scale.

Service de catalogue à haute disponibilité

Le domaine de services de catalogue est la grille des serveurs de catalogues que vous utilisez. Il contient les informations relatives à la topologie de tous les conteneurs de votre environnement eXtreme Scale. Le service de catalogue contrôle les fonctions d'équilibrage et de routage pour tous les clients. Pour déployer eXtreme Scale en tant qu'espace de traitement de la base de données en mémoire, vous devez regrouper le service de catalogue en un cluster de domaine de services de catalogue afin de garantir la haute disponibilité.

Composants du domaine de services de catalogue

Lorsque plusieurs serveurs de catalogues démarrent, l'un d'eux est choisi comme serveur maître acceptant les signaux de présence IIOP (Internet Inter-ORB Protocol) et gérant les modifications des données du système consécutives aux modifications apportées aux services de catalogue ou aux conteneurs.

Lorsque des clients contactent l'un des serveurs de catalogues, la table de routage du domaine de services de catalogue est propagée vers ces clients via le contexte de service CORBA (Common Object Request Broker Architecture).

Configurez au moins trois serveurs de catalogues. Si votre configuration contient des zones, vous pouvez configurer un serveur de catalogues par zone.

Lorsqu'un serveur conteneur eXtreme Scale contacte l'un des serveurs de catalogues, la table de routage du domaine de services de catalogue est également propagée au serveur conteneur eXtreme Scale via le contexte de service CORBA. En outre, si le serveur contacté n'est pas le serveur maître, la demande est automatiquement redirigée vers le serveur maître actuel et la table de routage du serveur de catalogues est mise à jour.

Remarque : La grille des serveurs de catalogues et la grille des serveurs conteneurs sont très différentes. La première, le domaine de services de catalogue, permet de garantir la haute disponibilité de vos données système. La seconde, la grille des conteneurs, permet d'assurer la haute disponibilité et l'extensibilité de vos données ainsi que la gestion des charges de travail. De ce fait, il existe deux différentes tables de routage : la table de routage du domaine de services de catalogue et celle des fragments de la grille des serveurs.

Les responsabilités du catalogue se répartissent en plusieurs sortes de services. Le gestionnaire du groupe central gère le regroupement homologue en vue de la surveillance de la santé, le service de positionnement prend en charge les allocations, le service d'administration fournit l'accès à l'administration et le service de localisation gère les localités.

Déploiement du domaine de services de catalogue

Gestionnaire du groupe central

Le service de catalogue utilise le gestionnaire de haute disponibilité (gestionnaire HA) pour regrouper des processus en vue de la surveillance de la disponibilité. Chaque regroupement est un groupe central. Avec eXtreme Scale, le gestionnaire du groupe central regroupe les processus de manière dynamique. Ces processus doivent être de petite taille afin de permettre l'évolutivité. Chaque groupe central choisit un responsable qui sera chargé d'envoyer un statut au gestionnaire du groupe central en cas de défaillance de certains membre. Le même mécanisme de

statut est utilisé pour identifier une éventuelle défaillance de tous les membres d'un groupe, qui pourrait entraîner un échec de la communication avec le responsable.

Le gestionnaire du groupe central est un service entièrement automatique responsable de l'organisation des conteneurs en petits groupes de serveurs qui sont alors automatiquement fédérés de manière souple pour constituer une grille d'objets. Lorsqu'un conteneur contacte le service de catalogue pour la première fois, il attend d'être affecté à un nouveau groupe ou à un groupe existant. Un déploiement eXtreme Scale consiste en plusieurs groupes de ce type et ce regroupement est une tâche d'activation essentielle pour l'évolutivité. Chaque groupe est un groupe de machines virtuelles Java utilisant les signaux de présence pour surveiller la disponibilité des autres groupes. L'un des membres de ce groupe est choisi comme responsable. Il est chargé de relayer les informations relatives à la disponibilité au service de catalogue afin de lui permettre de réagir aux défaillances en procédant à des réallocations et à des réacheminements.

Service de positionnement

Le service de catalogue gère le positionnement des fragments dans les conteneurs disponibles. Le service de positionnement est responsable du maintien de l'équilibre des ressources entre les différentes ressources physiques. Il prend également en charge l'allocation des fragments à leur conteneur hôte. Il s'exécute en tant que service Un sur N choisi dans la grille de données afin qu'il ne s'exécute qu'une instance et une seule de ce service. Si cette instance échoue, un autre processus est choisi et il prend le relais. Pour garantir la redondance, l'état du service de catalogue est répliqué sur tous les serveurs qui hébergent le service de catalogue.

Administration

Le service de catalogue constitue le point d'entrée logique de l'administration du système. Le service de catalogue héberge un bean géré (MBean) et il fournit des URL JMX (Java Management Extensions) pour les serveurs gérés par le service.

Service de localisation

Le service de localisation sert de point tactile pour les clients qui recherchent les conteneurs hébergeant l'application qu'ils recherchent, ainsi que pour les conteneurs qui enregistrent les applications hébergées auprès du service de positionnement. Il s'exécute sur tous les membres de la grille, permettant ainsi de supprimer cette fonction.

Le service de catalogue héberge une logique qui est généralement inactive lorsque l'état est stabilisé. Il a donc très peu d'incidence sur l'évolutivité. Il permet de gérer des centaines de conteneurs devenant disponibles simultanément. Configurez le service de catalogue dans une grille à des fins de disponibilité.

Planification

Une fois que le domaine de services de catalogue a démarré, ses membres se lient entre eux. La topologie du domaine de services de catalogue doit faire l'objet d'une planification prudente et méticuleuse, car il sera impossible de modifier la configuration du domaine au moment de l'exécution. Etendez la grille de manière aussi diversifiée que possible pour éviter d'éventuelles erreurs.

Démarrage d'un domaine de services de catalogue

Pour en savoir plus sur la création d'un domaine de services de catalogue, voir .

Connexion à un domaine autonome de services de catalogue des conteneurs eXtreme Scale intégrés à WebSphere Application Server

Vous pouvez configurer des conteneurs eXtreme Scale qui sont imbriqués dans un environnement WebSphere Application Server pour qu'ils se connectent à un domaine autonome de services de catalogue.

- 7.1+ Vous pouvez créer des domaines de services de catalogue dans la console d'administration. Voir .
-  (déprécié) Dans les précédentes versions, la connexion de services de catalogue à un domaine de services de catalogue s'effectuait par la création d'une propriété personnalisée. Cette propriété est toujours utilisable, mais son utilisation n'est pas encouragée car elle est considérée comme déprécié. Pour en savoir plus sur cette propriété personnalisée, voir dans le *Guide d'administration* les explications sur le démarrage du processus de service de catalogue dans un environnement WebSphere Application Server..

  Remarque : Collision des noms de serveur : cette propriété étant utilisée pour démarrer le serveur de catalogues eXtreme Scale et pour s'y connecter, un serveur de catalogues ne doit pas porter le même nom qu'un serveur WebSphere Application Server.

Pour plus d'informations sur les quorums de serveur de catalogues, consultez la rubrique du manuel *Présentation du produit*.

Quorums de serveurs de catalogue

Le quorum est le nombre minimum de serveurs de catalogue nécessaires à l'exécution des opérations de positionnement pour la grille. Le nombre minimum est l'ensemble complet des serveurs de catalogue à moins que le quorum n'ait été redéfini.

Termes importants à connaître

Voici, avec leur définition, une liste des termes utilisés à propos des quorums dans WebSphere eXtreme Scale.

- **Microcoupure** :: une microcoupure est une perte provisoire de connectivité entre un ou plusieurs serveurs.
- **Interruption totale** :: une interruption totale est une perte permanente de connectivité entre un ou plusieurs serveurs.
- **Centre de données** : : un centre de données est un groupe géographique de serveurs, connectés en générale par un réseau local (LAN).
- **Zone**: une zone est une option de configuration servant à regrouper des serveurs ayant en commun une caractéristique physique particulière. Exemples de zones regroupant des serveurs : centre de données (voir plus haut), réseau de zone, bâtiment, étage d'un bâtiment, etc.
- **Signal de présence** : : mécanisme permettant de déterminer si une machine virtuelle Java donnée est ou non en cours d'exécution.

Topologie

Nous allons expliquer le fonctionnement de WebSphere eXtreme Scale sur un réseau comprenant des composant non fiables (par exemple, un réseau constitué de plusieurs centres de données épars).

Espace des adresses IP

WebSphere eXtreme Scale nécessite un réseau dans lequel tout élément adressable puisse se connecter sans entraves à n'importe quel autre élément adressable du réseau. En d'autres termes, WebSphere eXtreme Scale requiert un espace de noms non hiérarchisé d'adresses IP. Il requiert également que tous les pare-feu autorisent la totalité du trafic à circuler entre les adresses IP et les ports utilisés par les machines virtuelles Java hébergeant des éléments de WebSphere eXtreme Scale.

Réseaux locaux connectés

A chaque réseau local est attribué un identificateur de zone pour les besoins de WebSphere eXtreme Scale. WebSphere eXtreme Scale tente agressivement de détecter les signaux de présence des machines virtuelles Java d'une même zone. Il suffit que l'un de ces signaux manque pour que se déclenche un événement de basculement si le service de catalogue a le quorum.

Domaine de services de catalogue et serveurs conteneurs

Un domaine de services de catalogue est une collection de machines virtuelles Java semblables. Un domaine de services de catalogue est une grille composée de serveurs de catalogue et il est de taille fixe. Or, le nombre des serveurs conteneurs, lui, est dynamique. Des serveurs conteneurs peuvent être ajoutés ou supprimés à la demande. Dans une configuration de trois centres de données, WebSphere eXtreme Scale nécessite une machine virtuelle Java de service de catalogue par centre de données.

Le domaine de services de catalogue utilise un mécanisme de quorum complet. Du fait de ce mécanisme de quorum complet, les membres de la grille doivent tous être d'accord sur n'importe quelle action à entreprendre.

Les machines virtuelles Java des serveurs conteneurs sont marquées avec un identificateur de zone. La grille des machines virtuelles Java de conteneurs est automatiquement fractionnée en petits groupes centraux de machines virtuelles Java. Un groupe central ne comprendra que des machines virtuelles Java de la même zone. Des machines virtuelles Java de zones différentes ne feront jamais partie du même groupe central.

Un groupe central essaiera agressivement de détecter la défaillance de l'une de ses machines virtuelles Java. Les machines virtuelles Java de conteneurs d'un groupe central ne doivent jamais s'étendre sur plusieurs réseaux locaux interconnectés comme dans un réseau étendu. Cela signifie qu'un groupe central ne peut avoir dans la même zone de conteneurs s'exécutant dans des centres de données différents.

Cycle de vie des serveurs

Démarrage des serveurs de catalogue

Les serveurs de catalogue sont démarrés à l'aide de la commande startOgServer. Les quorums sont désactivés par défaut. Pour les activer, deux possibilités : passer l'indicateur `-quorum enabled` dans la commande startOgServer ou ajouter la propriété `enableQuorum=true` dans le fichier des propriétés. Tous les serveurs de catalogue doivent avoir le même quorum.

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

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

Démarrage des serveurs conteneurs

Les serveurs conteneurs sont démarrés à l'aide de la commande startOgServer. Dans le cas d'une grille de plusieurs centres de données, les serveurs doivent utiliser l'étiquette zone pour identifier le centre de données dans lequel ils résident. Définir la zone sur les serveurs d'une grille permet à WebSphere eXtreme Scale de ne surveiller la bonne santé que des serveurs du centre de données, en réduisant le trafic entre les centres de données.

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

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

Arrêt des serveurs d'une grille

Les serveurs d'une grille sont arrêtés à l'aide de la commande stopOgServer. Lors de l'arrêt d'un centre de données entier pour des raisons de maintenance, passez la liste de tous les serveurs appartenant à cette zone. Cela permettra une transition d'état nette entre la zone en cours d'arrêt et la ou les zones qui continuent de fonctionner.

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

Détection des défaillances

WebSphere eXtreme Scale détecte les morts de processus via des événements de fermeture anormale de sockets. Le service de catalogue est immédiatement informé lorsqu'un processus prend fin. Les interruptions totales sont détectées via l'absence des signaux de présence. WebSphere eXtreme Scale se protège lui-même contre les microcoupures entre les centres de données en utilisant une implémentation de quorums.

Implémentation de la détection des signaux de présence

Nous allons expliquer comment la détection de l'activité est implémentée dans WebSphere eXtreme Scale.

Détection des signaux de présence des membres de groupes centraux

Le service de catalogue place des machines virtuelles Java de conteneurs dans des groupes centraux dont la taille est limitée. Pour détecter la défaillance de l'un de ses membres, un groupe central pourra s'y prendre de deux manières. Si le socket d'une machine virtuelle Java est fermé, cette machine virtuelle Java sera considérée comme morte. Par ailleurs, chaque membre émet un signal de présence sur ces sockets à une fréquence qui est déterminée par la configuration. Si une machine virtuelle Java ne réagit pas à ces signaux dans le délai maximum prévu par la configuration, elle sera considérée comme morte.

Un membre du groupe central est toujours désigné comme leader. Le leader du groupe central (CGL) est chargé d'informer régulièrement le service de catalogue que le groupe central fonctionne et de signaler toutes les modifications intervenues au sein du service de catalogue. La défaillance d'une machine virtuelle Java ou l'ajout au groupe d'une nouvelle machine sont considérés comme des modifications de groupe central.

Si le leader du groupe central ne parvient pas à contacter l'un des membres du domaine de services de catalogue, il continuera de réessayer.

Détection des signaux de présence du domaine de services de catalogue

Le domaine de services de catalogue ressemble à un groupe central privé dont les membres sont statiques avec un mécanisme de quorum. La détection des défaillances s'effectue de la même manière que pour un groupe central normal. La différence tient à la modification de son comportement puisqu'il inclut une logique de quorum. Le service de catalogue utilise également une configuration moins agressive pour la détection des signaux de présence.

Détection des signaux de présence des groupes centraux

Le service de catalogue a besoin de savoir quand des serveurs conteneurs sont défaillants. Chaque groupe central est chargé de déterminer les défaillances des machines virtuelles Java conteneurs et de les signaler au service de catalogue par l'intermédiaire du leader du groupe. La défaillance complète de la totalité des membres d'un groupe central est une éventualité qui peut arriver. Si la totalité du groupe central est défaillant, c'est au service de catalogue de détecter cette perte.

Si le service de catalogue marque une machine virtuelle Java conteneur comme défaillante et que le conteneur est ultérieurement signalé comme en fonctionnement, la machine virtuelle Java conteneur recevra l'ordre d'arrêter les serveurs conteneurs WebSphere eXtreme Scale. Une machine virtuelle Java dans cet état ne sera pas visible dans les requêtes xsadmin. Des messages dans les journaux de la machine virtuelle Java conteneur indiqueront que cette machine est tombée en panne. Vous devrez la redémarrer manuellement.

Si une perte de quorum s'est produite, la détection des signaux de présence sera suspendue en attendant le rétablissement du quorum.

Comportement du service de catalogue en matière de quorum

Normalement, les membres du service de catalogue disposent d'une connectivité pleine et entière. Le domaine de services de catalogue est un ensemble statique de machines virtuelles Java. WebSphere eXtreme Scale s'attend à ce que tous les membres du service de catalogue soient en permanence en ligne. Le service de catalogue ne répondra aux événements de conteneur que tant qu'il aura le quorum.

S'il perd le quorum, le service de catalogue attendra le rétablissement du quorum. Tant qu'il n'a pas le quorum, le service de catalogue ignorera les événements provenant des serveurs conteneurs. Ceux-ci retenteront pendant ce temps toutes les demandes rejetées par le serveur de catalogue puisque WebSphere eXtreme Scale eXtreme Scale s'attend à ce que le quorum soit rétabli.

Le message suivant indique que le quorum a été perdu. Recherchez la présence éventuelle de ce message dans les journaux de vos services de catalogue.

```
CWOBJ1254W: Le service de catalogue attend un quorum.
```

WebSphere eXtreme Scale s'attend à perdre le quorum pour les raisons suivantes :

- défaillance d'un membre machine virtuelle Java du service de catalogue
- microcoupure réseau
- perte de centre de données

Arrêter une instance de serveur de catalogue à l'aide de la commande `stopOgServer` ne provoque pas de perte de quorum car le système sait que l'instance a été arrêtée, ce qui n'est pas la même chose qu'une défaillance de machine virtuelle Java ou une microcoupure.

Perte de quorum due à une défaillance de machine virtuelle Java

La défaillance d'un serveur de catalogue provoquera la perte du quorum. Dans ce cas, le quorum doit être redéfini le plus vite possible. Le service de catalogue défaillant ne peut rejoindre la grille tant que le quorum n'a pas été redéfini.

Perte de quorum due à une microcoupure réseau

WebSphere eXtreme Scale est conçu pour intégrer la possibilité de microcoupures. Une microcoupure est une perte provisoire de connectivité entre des centres de données. Cette situation est transitoire par nature et en principe la situation est rétablie en quelques secondes ou en quelques minutes. WebSphere eXtreme Scale essaie de maintenir le bon fonctionnement pendant la microcoupure, celle-ci est donc considérée comme une simple défaillance. La défaillance est censée être réglée et le fonctionnement normal reprend sans intervention de WebSphere eXtreme Scale.

Une microcoupure qui s'éternise ne pourra être requalifiée en interruption totale que par une intervention d'utilisateur. La redéfinition du quorum sur l'un des côtés de la microcoupure est en effet nécessaire pour que l'événement soit classé comme interruption totale.

Cycle entre les services de catalogue

Si un serveur de catalogue est arrêté à l'aide de stopOgServer, le quorum diminue d'un serveur. Cela signifie que les serveurs restants ont toujours le quorum. Redémarrer le serveur de catalogue ramène le quorum au nombre précédent.

Conséquences de la perte de quorum

S'il arrive à une machine virtuelle Java conteneur de tomber en panne pendant la perte du quorum, la reprise n'aura pas lieu tant que l'on ne sera pas sorti de la microcoupure ou tant que l'utilisateur n'aura pas redéfini le quorum. WebSphere eXtreme Scale considère la perte de quorum et la défaillance d'un conteneur

comme une double défaillance, ce qui est un événement rare. Cela signifie que des applications peuvent perdre l'accès en écriture aux données qui étaient stockées sur la machine virtuelle Java défaillante jusqu'à ce que le quorum soit restauré, et ce n'est qu'alors que la reprise normale aura lieu.

De même, toute tentative de démarrage d'un conteneur pendant une perte de quorum est vouée à l'échec : le conteneur ne démarrera pas.

La connectivité clients complète est autorisée pendant la perte de quorum. S'il ne se produit aucune défaillance de conteneur ou de problèmes de connectivité pendant la perte du quorum, les clients pourront toujours interagir complètement avec les serveurs conteneurs.

Si une microcoupure se produit, certains clients risquent de pas pouvoir accéder aux copies primaires ou répliques des données jusqu'à la fin de la microcoupure.

Il est possible de démarrer de nouveaux clients, car il doit y avoir une machine virtuelle Java service de catalogue dans chaque centre de données, donc au moins une machine virtuelle service de catalogue sera accessible par le client même pendant une microcoupure.

Rétablissement du quorum

Si le quorum est perdu pour une quelconque raison, lorsqu'il est rétabli, un protocole de récupération est exécuté. Lorsque la perte du quorum se produit, toute vérification d'activité des groupes centraux est suspendue et les rapports signalant des défaillances sont ignorés. Une fois que le quorum est rétabli, le service de catalogue procède à une vérification de l'activité de tous les groupes centraux pour déterminer immédiatement leur composition. C'est à ce stade que seront récupérés tous les fragments précédemment hébergés sur des machines virtuelles Java signalées comme défaillantes. En cas de perte de fragments primaires, les fragments répliques survivants passent au statut de fragments primaires. En cas de perte de fragments répliques, des fragments répliques supplémentaires seront créées à partir des survivantes.

Redéfinir le quorum

Cela n'est à utiliser qu'en cas de défaillance d'un centre de données. Le quorum perdu suite à la défaillance d'une machine virtuelle Java service de catalogue ou à une microcoupure du réseau doit se rétablir automatiquement une fois la machine virtuelle Java redémarrée ou la microcoupure terminée.

Les administrateurs sont les seuls à être informés d'une défaillance du centre de données. WebSphere eXtreme Scale traite de manière semblable les microcoupures et les interruptions totales. Vous devez informer l'environnement eXtreme Scale de ces défaillances à l'aide de la commande xsadmin permettant de redéfinir le quorum. Le service de catalogue sera ainsi avisé de supposer que le quorum est atteint avec le nombre actuel de membres et la reprise complète aura lieu. En émettant une commande de redéfinition du quorum, vous garantissez que les machines virtuelles Java du centre de données défaillant sont réellement en panne et qu'elles ne reprendront pas leur activité.

La liste qui suit envisage quelques scénarios de redéfinition de quorum. Supposons que nous ayons trois serveurs de catalogue : A, B et C.

- Microcoupure : supposons que, suite à une microcoupure, C soit provisoirement isolé. Le service de catalogue perdra le quorum et attendra la fin de la

microcoupure. A ce moment-là, C réintégrera le domaine de services de catalogue et le quorum sera rétabli. Votre application ne percevra aucun problème pendant ce temps.

- Défaillance provisoire : ici, C est défaillant et le service de catalogue perd le quorum. Vous devez donc redéfinir ce dernier. Une fois le quorum rétabli, C pourra être redémarré. C réintégrera le domaine de services de catalogue lorsqu'il redémarrera. L'application ne percevra aucun problème pendant ce temps.
- Défaillance de centre de données : vous vérifiez que le centre de données est réellement défaillant et qu'il est bien isolé sur le réseau. Puis vous lancez la commande xsadmin de redéfinition de quorum. Les deux centres de données survivants procèdent à une reprise complète en remplaçant les fragments qui étaient hébergés dans le centre défaillant. Le service de catalogue s'exécute à présent avec un quorum complet de A et de B. L'application risque de constater des retards ou des exceptions pendant l'intervalle séparant le début de l'interruption totale et la redéfinition du quorum . Une fois ce dernier redéfini, la grille et le fonctionnement normal reprennent.
- Reprise du fonctionnement de centre de données : les centres de données survivants fonctionnent déjà avec un quorum redéfini. Lorsque le centre de données contenant C est redémarré, toutes les machines virtuelles Java du centre doivent être redémarrées. C réintégrera alors le domaine de services de catalogue existant et le quorum reviendra à la situation normale sans intervention de l'utilisateur.
- Défaillance de centre de données et microcoupure : le centre de données contenant C tombe en panne. Le quorum est redéfini et rétabli sur les centres de données restants. Si une microcoupure se produit entre A et B, les règles normales de reprise en cas de microcoupure s'appliquent. Une fois la microcoupure terminée, le quorum est rétabli et la récupération nécessaire après la perte du quorum se produit.

Comportement des conteneurs

Nous allons décrire comment se comporte les machines virtuelles Java des serveurs conteneurs pendant une perte et une récupération de quorum.

Les conteneurs hébergent un ou plusieurs fragments. Les fragments sont soit des fragments primaires, soit des fragments répliques pour une partition spécifique. Le service de catalogue affecte des fragments à un conteneur et le conteneur honorera cette affectation jusqu'à ce que nouvelles instructions arrivent du service de catalogue. Cela signifie que si, dans un conteneur, un fragment primaire ne parvient pas à communiquer avec un fragment réplique en raison d'une microcoupure, il continuera à réessayer jusqu'à ce qu'il reçoive de nouvelles instructions du service de catalogue.

Si une microcoupure se produit et qu'un fragment primaire perd la communication avec sa réplique, il continuera ses tentatives de connexion jusqu'à ce que le service de catalogue lui envoie de nouvelles instructions.

Comportement des fragments répliques synchrones

Pendant que la connexion est interrompue, le fragment primaire peut accepter de nouvelles transactions tant que le nombre de fragments répliques en ligne est au moins égal à la valeur définie pour la propriété minsync du groupe de mappes. Si de nouvelles transactions sont traitées sur le fragment primaire pendant que la

liaison avec le fragment réplique synchrone est interrompue, le fragment réplique synchrone sera effacé et resynchronisé avec l'état actuel du fragment primaire lorsque la liaison sera rétablie.

La réplication synchrone est fortement découragée entre les centres de données ou sur les liaisons de type réseau étendu.

Comportement des fragments répliques asynchrones

Pendant que la connexion est interrompue, le fragment primaire peut accepter de nouvelles transactions. Le fragment primaire mettra les modifications en mémoire tampon jusqu'à une certaine limite. Si la connexion au fragment réplique est rétablie avant que cette limite ne soit atteinte, le fragment réplique sera actualisé avec les modifications mises en mémoire tampon. Si la limite est atteinte, le fragment primaire détruit la liste en tampon et, lorsqu'il se reconnecte, le fragment réplique est effacé et resynchronisé.

Comportement des clients

Les clients sont toujours en mesure de se connecter au serveur de catalogue pour s'amorcer sur la grille que le domaine de services de catalogue ait ou non le quorum. Le client essaiera de se connecter à n'importe quelle instance de serveur de catalogue pour obtenir une table de routage afin d'interagir avec la grille. La connectivité réseau peut empêcher le client d'interagir avec certaines partitions en raison de la configuration du réseau. Le client peut se connecter aux répliques locales des données distantes s'il a été configuré pour cela. Les clients ne seront pas en mesure d'actualiser des données si la partition primaire de ces données n'est pas disponible.

Commandes xsadmin de quorum

Nous allons passer en revue les commandes xsadmin utiles pour les quorums.

Interroger le statut du quorum

Il est possible d'interroger avec la commande xsadmin le statut du quorum d'une instance de serveur de catalogue.

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

Cinq résultats sont possibles.

- Le quorum est désactivé : les serveurs de catalogue s'exécutent en mode quorum-disabled. L'on est en mode développement ou en mode centre de données unique. Ce n'est pas recommandé pour les configurations impliquant plusieurs centres de données.
- Le quorum est activé et le serveur de catalogue a le quorum : le quorum est activé et le système fonctionne normalement.
- Le quorum est activé et le serveur de catalogue attend le quorum : le quorum est activé et il a été perdu.
- Le quorum est activé et il est redéfini : le quorum est activé et il a été redéfini.
- Le quorum est proscrit : lorsqu'une microcoupure se produit, le service de catalogue est scindé en deux partitions, A et B. Le serveur de catalogue A a un quorum qui a été redéfini. La partition réseau se résout et le serveur dans la partition B est proscrit, nécessitant un redémarrage des machines virtuelles Java.

Cela se produit également si la machine virtuelle Java du catalogue redémarre pendant la microcoupure et que cette dernière se termine.

Redéfinir le quorum

La commande xsadmin peut servir à redéfinir un quorum. Toutes les instances survivantes de serveurs de catalogue sont utilisables. Tous les survivants sont notifiés lorsque l'un d'entre eux reçoit l'injonction de redéfinir le quorum. La syntaxe est la suivante.

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

Commandes de diagnostics

- Statut du quorum : voir la section précédente.
- Liste des groupes centraux : affiche la liste de tous les groupes centraux. Les membres et les leaders des groupes centraux sont affichés.

  ```
  xsadmin –ch cathost –p 1099 –coregroups
  ```
- Arrêt de serveurs : cette commande retire manuellement un serveur de la grille. Ce n'est en principe pas nécessaire puisque les serveurs sont automatiquement retirés lorsqu'ils sont détectés comme défaillants, mais la commande est quand même fournie pour être utilisée sous les directives du support IBM.

  ```
  xsadmin –ch cathost –p 1099 –g Grid –teardown server1,server2,server3
  ```
- Affichage de la table de routage : cette commande affiche la table de routage en cours en simulant une nouvelle connexion client à la grille. Elle valide également la table de routage en confirmant que tous les serveurs conteneurs reconnaissent bien leur rôle dans la table (par exemple, quel type de fragment pour quelle partition).

  ```
  xsadmin –ch cathost –p 1099 –g myGrid -routetable
  ```
- Affichage des fragments non attribués : si certains fragments ne peuvent être placés dans la grille, cette commande permet d'afficher leur liste. Cela ne se produit que lorsque le service de positionnement comporte une contrainte qui empêche le positionnement. Exemple : si, en en mode de production, vous démarrez des machines virtuelles Java sur un seul système, seuls seront placés les fragments primaires. Les fragments répliques ne seront attribués que lorsque des machines virtuelles Java démarreront sur un second système. Le service de positionnement ne place des fragments répliques que sur des machines virtuelles Java dont les adresses IP sont différentes de celles des machines virtuelles Java hébergeant les fragments primaires. Ne pas avoir de machines virtuelles Java dans une zone peut également provoquer la non-attribution de fragments.

  ```
  xsadmin –ch cathost –p 1099 –g myGrid –unassigned
  ```
- Paramétrer la trace : cette commande définit les paramètres de trace pour toutes les machines virtuelles Java correspondant au filtre spécifié pour la commande xsadmin. Les paramètres de trace ne sont modifiés que jusqu'à ce qu'une autre commande soit utilisée ou que les machines virtuelles Java tombent en panne ou s'arrêtent.

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

  La trace est activée sur toutes les machines virtuelles Java du système dont le nom d'hôte est spécifiée (host1 ici).
- Vérification de la taille des mappes : la commande mapsizes est utile pour vérifier que la répartition des clés est uniforme sur les fragments présents dans la clé. Si certains conteneurs détiennent un nombre de clés significativement plus

important que d'autres, c'est l'indice que la fonction de hachage sur les objets key a une répartition de mauvaise qualité.

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

Considérations relatives à la sécurisation des transports

Les centres de données étant normalement déployés dans des sites géographiquement dispersés, les utilisateurs voudront activer la sécurité des transports entre ces centres.

Lire les explications concernant la sécurité des couches de transport dans le *Guide d'administration*.

Liste de contrôle opérationnelle

Utilisez la liste de contrôle opérationnelle afin de préparer votre environnement pour le déploiement de WebSphere eXtreme Scale.

Tableau 6. Liste de contrôle opérationnelle

Elément de la liste de contrôle	Pour plus d'informations

Si vous utilisez AIX, optimisez les paramètres suivants du système d'exploitation :
 	
TCP_KEEPINTVL
Le paramètre TCP_KEEPINTVL fait partie d'un protocole de maintien de connexion du socket qui permet la détection des indisponibilités du réseau. La propriété spécifie l'intervalle entre les paquets envoyés pour valider la connexion. Si vous utilisez WebSphere eXtreme Scale, spécifiez la valeur `10`. Pour vérifier le paramètre actuel, exécutez la commande suivante :
`# no –o tcp_keepintvl`
Pour modifier le paramètre actuel, exécutez la commande suivante :
`# no –o tcp_keepintvl=10`
Le paramètre TCP_KEEPINTVL est exprimé en demi secondes.
 	
TCP_KEEPINIT
Le paramètre TCP_KEEPINIT fait partie d'un protocole de maintien de connexion du socket qui permet la détection des indisponibilités du réseau. La propriété spécifie le délai d'attente initial de la connexion TCP. Si vous utilisez WebSphere eXtreme Scale, spécifiez la valeur `40`. Pour vérifier le paramètre actuel, exécutez les commandes suivantes :
`# no –o tcp_keepinit`
Pour modifier le paramètre actuel, exécutez la commande suivante :
`# no –o tcp_keepinit=40`
Le paramètre TCP_KEEPINIT est exprimé en demi secondes.
	• Pour des informations sur l'optimisation d'AIX, voir la rubrique Optimisation des systèmes AIX.

Mettez à jour le fichier `orb.properties` pour modifier le comportement de transport de la grille. Le fichier `orb.properties` se trouve dans le répertoire `java/jre/lib`.	«Fichier de propriétés de l'ORB», à la page 195

Tableau 6. Liste de contrôle opérationnelle (suite)

Elément de la liste de contrôle	Pour plus d'informations
Utilisez les paramètres du script startOgServer. En particulier, utilisez les paramètres suivants : • Définissez les paramètres de segment de mémoire avec l'option **-jvmArgs**. • Définissez les propriétés et le chemin d'accès aux classes avec l'option **-jvmArgs**. • Définissez les paramètres **-jvmArgs** pour configurer la surveillance de l'agent. **Paramètres de port** WebSphere eXtreme Scale doit ouvrir des ports pour les communications pour certains transports. Ces ports sont tous définis de manière dynamique. Toutefois, si un pare-feu est utilisé entre les conteneurs, vous devez spécifier les ports. Utilisez les informations suivantes sur les ports : **Port du programme d'écoute** Vous pouvez utiliser l'argument **-listenerPort** pour spécifier le port utilisé pour les communications entre les processus. **Port du groupe central** Vous pouvez utiliser l'argument **-haManagerPort** pour spécifier le port utilisé pour la détection des incidents. Cet argument correspond à peerPort. Notez que les groupes centraux n'ayant pas besoin de communiquer entre les zones, il n'est pas nécessaire de définir ce port si le pare-feu est ouvert pour tous les membres d'une même zone. **Port du service JMX** Vous pouvez utiliser l'argument **-JMXServicePort** pour spécifier le port à utiliser par le service JMX. **Port SSL** Si vous transmettez `-Dcom.ibm.CSI.SSLPort=1234` comme argument **-jvmArgs**, le port SSL prend la valeur `1234`. Le port SSL est l'homologue de port sécurisé du port du programme d'écoute. **Port du client** Utilisé uniquement dans le service de catalogue. Vous pouvez spécifier cette valeur avec l'argument **-catalogServiceEndPoints**. La valeur de ce paramètre est au format suivant : `nomServeur:nomHôte:portClient:portHomologue`	«Script startOgServer», à la page 336
Vérifiez que les paramètres de sécurité sont configurés correctement : • Transport (SSL) • Application (Authentification et autorisation) Pour vérifier vos paramètres de sécurité, vous pouvez essayer d'utiliser un client malveillant pour vous connecter à votre configuration. Par exemple, si le paramètre SSL requis est configuré, un client possédant un paramètre TCP_IP avec ce dernier ou un client possédant un fichier de clés certifiées incorrect ne doit pas pouvoir se connecter au serveur. Si l'authentification est requise, un client sans données d'identification, telles qu'un ID utilisateur et un mot de passe ne doit pas pouvoir se connecter au serveur. Si l'autorisation est appliquée, un client sans autorisation d'accès ne doit pas être autorisé à accéder aux ressources du serveur.	«Intégration de la sécurité à des fournisseurs externes», à la page 370
Choisissez comment vous allez surveiller votre environnement. • xsAdmin – Les ports JMX des serveurs de catalogues doivent être visibles par l'outil XSAdmin. Les ports de conteneur doivent également être accessibles pour certaines commandes qui collectent des informations des conteneurs. • Vous pouvez choisir entre les outils de surveillance de fournisseur suivants : – Tivoli Enterprise Monitoring Agent – CA Wily Introscope – Hyperic HQ	• «Surveillance à l'aide de l'exemple d'utilitaire xsAdmin», à la page 387 • «Sécurité JMX (Java Management Extensions)», à la page 368 • «Surveillance à l'aide d'IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale», à la page 410 • «Surveillance d'eXtreme Scale à l'aide de Hyperic HQ», à la page 420 • «Surveillance des applications eXtreme Scale à l'aide de CA Wily Introscope», à la page 416

Chapitre 6. Configuration de l'environnement de déploiement

Vous pouvez configurer WebSphere eXtreme Scale pour qu'il soit exécuté dans un environnement autonome ou configurer eXtreme Scale pour qu'il soit exécuté dans un environnement avec WebSphere Application Server ou WebSphere Application Server Network Deployment. Pour qu'un déploiement eXtreme Scale extrait les changements de configuration côté grille du serveur, vous devez redémarrer les processus pour appliquer ces changements au lieu d'attendre qu'ils soient appliqués de manière dynamique. Toutefois, côté client, vous ne pouvez pas modifier les paramètres de configuration d'une instance de client existante, mais vous pouvez créer un client avec les paramètres dont vous avez besoin à l'aide d'un fichier XML ou d'un programme. Lorsque vous créez un client, vous pouvez remplacer les paramètres par défaut provenant de la configuration de serveur actuelle.

Méthodes de configuration

Les fichiers XML et les fichiers de propriétés sont les méthodes les plus usuelles de configuration du produit lorsqu'on ne recourt pas à la programmation. Vous trouverez dans le Guide de programmation des explications concernant les autres méthodes : interfaces de programmation de système et d'applications, plug-in et beans gérés.

Configuration à l'aide de fichiers XML

WebSphere eXtreme Scale est configuré par une collection de fichiers XML.

Les configurations suivantes d'eXtreme Scale peuvent être effectuées grâce à des fichiers XML :

- Règle de déploiement – Pour configurer une règle de déploiement, utilisez un fichier XML de descripteur de règle de déploiement. La rubrique Fichier XML du descripteur de la règle de déploiement explique comment définir les éléments et les attributs de ce fichier en fonction de divers besoins.
- Descripteur d'ObjectGrid - Pour configurer les détails des instances individuelles d'ObjectGrids, utilisez un fichier XML de descripteur. Lire la rubrique concernant «Fichier XML du descripteur d'ObjectGrid», à la page 143.
- Configurer des entités - En fonction de vos besoins en entités dans votre déploiement tel qu'il est défini dans un schéma logique, vous pouvez configurer ces entités à l'aide de classes Java annotées, de XML ou d'une combinaison des deux. Les entités définies sont ensuite enregistrées auprès d'un serveur eXtreme Scale et liées à des mappes de sauvegarde, des index et d'autres plug-in. Un schéma d'entité est un ensemble d'entités et des relations entre ces entités. La rubrique «Configuration d'entités», à la page 217 expose de manière détaillée des options de configuration permettant d'optimiser des schémas d'entités.
- Configurer la sécurité - Vous pouvez activer à l'aide de fichiers XML de configuration la sécurité pour un déploiement particulier. La section «Fichier XML du descripteur de sécurité», à la page 375 expose les possibilités de configuration en matière d'activation de la sécurité.
- Configurer des clients - Vous pouvez utiliser un fichier de propriétés et d'autres méthodes pour spécifier les noms d'hôtes des clients, leurs ports, leur sécurité, et

d'autres informations. La section «Configuration des clients», à la page 204 explique plus en détail comment personnaliser des clients à l'aide d'un fichier XML.

- Configuration de l'intégration à Spring - Un certain nombre de méthodes vous permettent d'activer la synergie du cadre applicatif Spring avec un environnement de déploiement d'eXtreme Scale : scripts XML et définition de schéma, assortis d'autres méthodes comme les beans d'extension et la prise en charge des espaces de noms. La rubrique «Configuration de l'intégration à Spring», à la page 276 explique comment utiliser eXtreme Scale avec Spring.

Identification et résolution des problèmes concernant la configuration XML

Au cours de la configuration d'eXtreme Scale, vous pouvez rencontrer occasionnellement un comportement inattendu dans la configuration XML.

Les sections suivantes répertorient les divers problèmes de configuration XML que vous pouvez rencontrer.

Problème de concordance entre la règle de déploiement et les fichiers XML ObjectGrid

La règle de déploiement et les fichiers XML ObjectGrid doivent concorder. Si les noms ObjectGrid et les noms de la mappe ne concordent pas, des erreurs se produisent.

backingMap et références de mappe incorrectes

Si la liste de la backingMap dans un fichier XML ObjectGrid ne correspond pas à la liste des références de la mappe d'un fichier XML de règle de déploiement, une erreur se produit sur le serveur de catalogue.

Par exemple, le fichier XML ObjectGrid et le fichier XML de la règle de déploiement ci-dessous permettent de démarrer un processus de conteneur. Le fichier de la règle de déploiement contient davantage de références de mappe que celles listées dans le fichier XML ObjectGrid.

ObjectGrid.xml - exemple incorrect

```
<?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 - exemple incorrect

```
<?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>
```

Messages

Un message d'erreur est écrit dans le fichier SystemOut.log lorsque la règle de déploiement est incompatible avec le fichier XML ObjectGrid. Pour l'exemple précédent, ce message est le suivant :

```
CWOBJ3179E: La référence à la mappe Ledger dans le groupe de mappes mapSet1 du fichier de descripteur
de déploiement Accouting ne fait pas référence à une mappe de
sauvegarde valide dans le XML de l'ObjectGrid.
```

S'il manque dans la règle de déploiement des références aux backingMaps répertoriées dans le fichier XML ObjectGrid, un message d'erreur est écrit dans le fichier SystemOut.log. Par exemple :

```
CWOBJ3178E: La mappe Ledger dans Accounting de l'ObjectGrid référencée dans le XML de l'ObjectGrid est
introuvable dans le fichier de descripteur du déploiement.
```

Incident

La liste des backingMaps dans le fichier XML ObjectGrid et les références de mappe de la règle de déploiement doivent correspondre.

Solution

Déterminez quelle est la liste adéquate pour votre environnement et modifiez le fichier XML contenant la liste incorrecte.

Noms ObjectGrid incorrects

Le nom de l'ObjectGrid est référencé à la fois dans le fichier XML ObjectGrid et le fichier XML de la règle de déploiement.

Message

Une exception ObjectGridException s'est produite avec l'exception IncompatibleDeploymentPolicyException. Voici un exemple.

Motif : com.ibm.websphere.objectgrid.IncompatibleDeploymentPolicyException : l'objectgridDeployment avec l'objectGridName accountin n'a pas d'objectGrid correspondant dans le fichier XML ObjectGrid.

Incident

Le fichier XML ObjectGrid est la liste principale des noms ObjectGrid. Si une règle de déploiement a un nom ObjectGrid qui n'est pas contenu dans le fichier XML ObjectGrid, une erreur se produit.

Solution

Vérifiez l'orthographe du nom ObjectGrid. Supprimez les noms redondants ou ajoutez les noms ObjectGrid manquants dans le fichier XML ObjectGrid ou le fichier XML de la règle de déploiement. Dans l'exemple de message, la valeur de l'objectGridName est mal orthographiée ("accountin" au lieu de "accounting").

La valeur XML de l'attribut n'est pas valide

Certains attributs du fichier XML acceptent un nombre limité de valeurs possibles. Les valeurs acceptées par ces attributs sont énumérées par le schéma. La liste suivante indique certains de ces attributs :

- Attribut authorizationMechanism sur l'élément objectGrid
- Attribut copyMode sur l'élément backingMap

- Attribut lockStrategy sur l'élément backingMap
- Attribut ttlEvictorType sur l'élément backingMap
- Attribut type sur l'élément property
- initialState sur l'élément objectGrid
- evictionTriggers sur l'élément backingMap

Si une valeur non valide est attribuée à l'un de ces attributs, la validation XML échoue. Dans l'exemple suivant de fichier XML, une valeur incorrecte INVALID_COPY_MODE est utilisée :

Exemple 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>
```

Le message suivant s'affiche dans le fichier journal.

```
CWOBJ2403E : le fichier XML n'est pas valide. Un problème a été détecté avec
< null > à la ligne 5. Le message d'erreur est
le suivant : cvc-enumeration-valid :
la valeur 'INVALID_COPY_MODE' n'est pas valide pour la
facette par rapport à l'énumération
'[COPY_ON_READ_AND_COMMIT, COPY_ON_READ, COPY_ON_WRITE,
NO_COPY, COPY_TO_BYTES]'.
La valeur doit être l'une des valeurs énumérées.
```

Balises ou attributs manquants

Si un attribut ou une balise manque dans un fichier XML, des erreurs se produisent. Par exemple, la balise fermante < /objectGrid > manque dans le fichier XML ObjectGrid suivant :

attributs manquants- exemple XML

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

   </objectGrids>
</objectGridConfig>
```

Le message suivant s'affiche dans le fichier journal.

```
CWOBJ2403E : le fichier XML n'est pas valide. Un problème a été détecté avec
< null > à la ligne 7. Le message d'erreur est le suivant :
la balise de fin du type d'élément "objectGrid" doit se
terminer avec le délimiteur '>'.
```

Une exception ObjectGridException concernant le fichier XML non valide se produit avec le nom du fichier XML.

Erreurs de syntaxe

Si la syntaxe du fichier XML est incorrecte, CWOBJ2403E apparaît dans le fichier journal. Par exemple, le message suivant s'affiche lorsqu'un guillemet est manquant pour l'un des attributs XML.

```
CWOBJ2403E : le fichier XML n'est pas valide. Un problème a été détecté
avec < null > à la ligne 7. Le message d'erreur est le suivant :
un guillemet ouvrant est attendu pour l'attribut "maxSyncReplicas"
associé à un type d'élément "mapSet".
```

Une exception ObjectGridException concernant le fichier XML non valide se produit également.

Référencement d'une collection de plug-in inexistante

Lorsque vous utilisez XML pour définir des plug-in BackingMap, l'attribut pluginCollectionRef de l'élément backingMap doit faire référence à une backingMapPluginCollection. L'attribut pluginCollectionRef doit correspondre à l'ID de l'un des éléments de la backingMapPluginCollection.

Message

Si l'attribut pluginCollectionRef ne correspond à aucun attribut d'ID de l'un des éléments backingMapPluginConfiguration, le message suivant ou un message similaire s'affiche dans le fichier journal.

```
[7/14/05 14:02:01:971 CDT] 686c060e XmlErrorHandl E CWOBJ9002E :
Message informatif en anglais uniquement : fichier XML non valide. Ligne : 14 ;
URI : null ; Message : la clé 'pluginCollectionRef' avec
la valeur 'bookPlugins' est introuvable pour la contrainte d'identité de
l'élément 'objectGridConfig'.
```

Incident

Le fichier XML suivant est utilisé pour produire l'erreur. Notez que l'attribut pluginCollectionRef de la BackingMap book est défini sur bookPlugins et que l'ID de la backingMapPluginCollection est collection1.

Référencement d'un attribut XML inexistant - Exemple

```
<?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>
```

Solution

Pour résoudre le problème, assurez-vous que la valeur de chaque pluginCollectionRef correspond à l'ID de l'un des éléments de la

backingMapPluginCollection. Modifiez simplement le nom de pluginCollectionRef en collection1 pour ne pas recevoir cette erreur. Une autre solution consiste à modifier l'ID de la backingMapPluginCollection existante pour le faire correspondre à pluginCollectionRef, ou d'ajouter une backingMapPluginCollection ayant un ID correspond à pluginCollectionRef.

Validation de XML sans implémentation

La version 1.4.2 du kit de développement de logiciels IBM contient l'implémentation de certaines fonctions de Java API for XML Processing (JAXP) pour utiliser la validation XML avec un schéma.

Lorsque vous utilisez un kit de développement de logiciels ne contenant pas cette implémentation, les tentatives de validation risquent d'échouer. Si vous souhaitez valider XML en utilisant un kit de développement de logiciels ne contenant pas cette implémentation, téléchargez Apache Xerces, et incluez les fichiers Java archive (JAR) dans le chemin d'accès aux classes.

Lorsque vous tentez de valider XML avec un kit de développement de logiciels n'ayant pas l'implémentation nécessaire, le fichier journal contient l'erreur suivante :

```
La validation XML XmlConfigBuild est activée
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 : aucun attribut implémenté
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)...
```

Le kit de développement de logiciels utilisé ne contient pas l'implémentation de la fonction JAXP nécessaire pour valider les fichiers XML en fonction d'un schéma.

Pour éviter ce problème, après avoir téléchargé Xerces et inclus les fichiers JAR dans le chemin d'accès aux classes, vous pouvez valider le fichier XML.

Référence de fichier de propriétés

Les fichiers de propriétés serveur contiennent les paramètres d'exécution de vos serveurs de catalogues et serveurs conteneurs. Vous pouvez spécifier un fichier de propriétés serveur pour une configuration autonome ou WebSphere Application Server. Les fichiers de propriétés client contiennent les paramètres de votre client.

Exemples de fichier de propriétés

Vous pouvez utiliser les exemples de fichier de propriétés suivants du répertoire *racine_extremescale*\properties pour créer vos fichiers de propriétés :

- sampleServer.properties
- sampleClient.properties

Propriétés système dépréciées

-Dcom.ibm.websphere.objectgrid.CatalogServerProperties
Cette propriété est dépréciée depuis la version 7.0 de WebSphere eXtreme Scale. Utilisez la propriété **-Dobjectgrid.server.props**.

-Dcom.ibm.websphere.objectgrid.ClientProperties
Cette propriété est dépréciée depuis la version 7.0 de WebSphere eXtreme Scale. Utilisez la propriété **-Dobjectgrid.client.props**.

-Dobjectgrid.security.server.prop
Cette propriété est dépréciée depuis la version 6.1.0.3 de WebSphere eXtreme Scale. Utilisez la propriété **-Dobjectgrid.server.prop**.

-serverSecurityFile
Cet argument n'est plus utilisé dans WebSphere eXtreme Scale Version 6.1.0.3. Cette option est transmise dans le script `startOgServer`. Utilisez l'argument **-serverProps**.

Configuration de grilles

Un fichier XML descripteur d'ObjectGrid permet de configurer des grilles, des mappes de sauvegarde, des plug-in, etc. Pour configurer WebSphere eXtreme Scale, utilisez un fichier XML de descripteur d'ObjectGrid ainsi que l'API ObjectGrid. Dans le cas d'une topologie répartie, vous aurez non seulement besoin d'un fichier XML de descripteur d'ObjectGrid, mais d'un fichier XML de règle de déploiement.

Configuration de déploiements locaux

Une configuration eXtreme Scale local en mémoire peut être créée à l'aide d'un fichier XML de descripteur d'ObjectGrid ou d'API eXtreme Scale.

Pourquoi et quand exécuter cette tâche

Le fichier `companyGrid.xml` ci-après est un exemple de XML de descripteur d'ObjectGrid. Les premières lignes de ce fichier incluent l'en-tête requis de chaque fichier XML ObjectGrid. Le fichier définit une instance ObjectGrid nommée "CompanyGrid" et plusieurs mappes de sauvegarde intitulées "Customer," "Item," "OrderLine" et "Order."

fichier
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>
```

Transmettez le fichier XML à l'une des méthodes createObjectGrid de l'interface ObjectGridManager. L'exemple de code ci-après valide le fichier `companyGrid.xml` par rapport au schéma XML et crée l'instance ObjectGrid intitulée "CompanyGrid." L'instance ObjectGrid nouvellement créée n'est pas placée en cache.

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

Vosu pouvez également créer des instances ObjectGrid à l'aide d'un programme, sans XML. Par exemple, vous pouvez utiliser le fragment de code ci-après à la place du code et du XML précédents.

```
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");
```

Pour une description complète du fichier XML d'ObjectGrid, reportez-vous à la eXtreme Scale référence de configuration.

Configuration du plug-in HashIndex

Le HashIndex classe com.ibm.websphere.objectgrid.plugins.index.HashIndex pré-intégrée) est un plug-in MapIndexPlugin qu'il est possible d'ajouter à une BackingMap pour générer des index statiques ou dynamiques. Il prend en charge les deux interfaces MapIndex et MapRangeIndex. Des index judicieusement définis et utilisés ne peuvent qu'améliorer considérablement les performances des requêtes.

Pour plus d'informations sur l'indexation, voir Indexation et HashIndex composite. Pour savoir comment utiliser l'indexation, voir Utilisation de l'indexation pour l'accès aux données ne correspondant pas à une clé et HashIndex composite.

Attributs permettant de configurer HashIndex

Les attributs suivants peuvent servir à configurer le plug-in HashIndex, soit dans un fichier XML de descripteur de déploiement ObjectGrid, soit dans le code d'un programme :

Name Spécifie le nom de l'index. Le nom doit être unique pour chaque mappe. Il sert à extraire l'objet d'index de l'instance ObjectMap pour la mappe de sauvegarde.

AttributeName
Spécifie à l'index les noms des attributs, séparés par des virgules. Pour les index d'accès par champ, les noms d'attributs sont équivalents aux noms des champs. Pour les index d'accès par propriété, les noms d'attributs sont les noms de propriétés compatibles JavaBean. En cas de nom d'attribut unique, l'index HashIndex est un index d'attribut unique et si cet attribut est une relation, il est également un index de relation. Si les noms d'attributs recouvrent plusieurs attributs, l'index HashIndex sera un index composite.

FieldAccessAttribute
Utilisé pour les mappes qui ne correspondent pas à des entités. Si la valeur est égale à `true`, l'accès à l'objet s'effectue par les champs. Si la valeur n'est pas spécifiée ou si elle est `false`, c'est la méthode get de l'attribut qui servira à accéder aux données.

POJOKeyIndex
Utilisé pour les mappes qui ne correspondent pas à des entités. Si la valeur est `true`, l'index introspectera l'objet dans la partie clé de la mappe. Cet attribut est utile lorsque la clé est une clé composite et que la valeur n'intègre pas la clé. Si la valeur n'est pas spécifiée ou si elle est égale à `false`, l'index introspectera l'objet dans la partie valeur de la mappe.

RangeIndex

Si la valeur est égale à `true`, l'indexation de plage est activée et l'application peut transtyper vers l'interface MapRangeIndex l'objet d'index extrait. Si la propriété RangeIndex est configurée comme `false`, l'application ne pourra transtyper l'objet d'index extrait que vers l'interface MapIndex.

Ajouter un HashIndex à une BackingMap

Plusieurs approches sont possibles pour ajouter un HashIndex à une BackingMap. L'exemple suivant illustre comment ajouter par configuration XML des plug-in d'indexation statiques :

Configurer par XML l'ajout de HashIndex à une BackingMap

```
<backingMapPluginCollection id="person">
   <bean id="MapIndexplugin"
      className="com.ibm.websphere.objectgrid.plugins.index.HashIndex">
         <property name="Name" type="java.lang.String" value="CODE"
            description="index name" />
         <property name="RangeIndex" type="boolean" value="true"
            description="true for MapRangeIndex" />
         <property name="AttributeName" type="java.lang.String" value="employeeCode"
            description="attribute name" />
   </bean>
</backingMapPluginCollection>
```

Dans cet exemple de configuration XML, la classe pré-intégrée HashIndex est utilisée comme plug-in d'indexation. HashIndex prend en charge des propriétés que l'utilisateur peut configurer telles que Name, RangeIndex et AttributeName dans l'exemple précédent.

- La propriété Name est configurée comme `CODE`, une chaîne identifiant ce plug-in d'indexation. La valeur de la propriété Name doit être unique dans l'étendue de la mappe de sauvegarde et peut servir à extraire l'objet d'index de l'instance ObjectMap pour la mappe de sauvegarde.
- La propriété RangeIndex est configurée avec la valeur `true`, ce qui signifie que l'application peut transtyper vers l'interface MapRangeIndex l'objet d'index extrait. Si la propriété RangeIndex est configurée comme `false`, l'application ne pourra transtyper l'objet d'index extrait que vers l'interface MapIndex. Une interface MapRangeIndex prend en charge des fonctions de recherche de données à l'aide des fonctions de plage (range) telles que greater than, less than ou les deux, alors qu'un index MapIndex prend uniquement en charge les fonctions d'égalité (equals). Si l'index est utilisé par la requête, la propriété RangeIndex doit avoir la valeur `true` dans les index à attribut unique. Dans le cas d'un index de relations ou d'un index composite, la propriété RangeIndex doit être configurée comme `false`.
- La propriété AttributeName est configurée avec la valeur `employeeCode`, ce qui signifie que l'attribut employeeCode de l'objet mis en cache servira à générer un index à attribut unique. Si une application doit rechercher des objets mis en cache pour plusieurs attributs, la propriété AttributeName peut être définie sur une liste d'attributs délimitée par des virgules, produisant un index composite.

Pour résumer, l'exemple précédent définit un HashIndex de plage à attribut unique. HashIndex à attribut unique, c'est également un HashIndex de plage.

HashIndex à attribut unique ou HashIndex composite ?

Si la propriété AttributeName de l'index HashIndex contient plusieurs noms d'attributs, l'index HashIndex est composite. Dans le cas contraire, si l'index inclut un seul nom d'attribut, il s'agit d'un index à attribut unique. Par exemple, la valeur de la propriété AttributeName d'un HashIndex composite pourra être `city,state,zipcode`. L'index contient trois attributs séparés par des virgules. Si la valeur de la propriété AttributeName est uniquement `zipcode` (un seul attribut), l'index HashIndex sera un index à attribut unique. L'exemple qui précède est un HashIndex à attribut unique car la propriété AttributeName a pour valeur “employeeCode”, laquelle ne comprend qu'un seul nom d'attribut.

Les index HashIndex composites constituent un mode efficace de consultation des objets mis en cache lorsque les critères de recherche impliquent plusieurs attributs. Toutefois, ils ne prennent pas en charge les index de plage et la propriété RangeIndex correspondante doit être définie sur false.

Reportez-vous à la rubrique relative aux index composites dans le *guide d'administration*.

HashIndex de relation

Si l'attribut indexé d'un HashIndex à attribut unique est une relation, que ce soit à valeur unique ou à valeurs multiples, l'index HashIndex est un HashIndex de relation. Pour l'index HashIndex de relation, la propriété RangeIndex de l'index HashIndex doit être définie sur false.

Les HashIndex de relation peuvent accélérer l'exécution des requêtes qui exploitent des références cycliques ou qui utilisent les filtres de requête IS NULL, IS EMPTY, SIZE et MEMBER OF. Reportez-vous à la rubrique relative à l'optimisation des requêtes à l'aide des index dans le *guide de programmation*.

HashIndex de clés

Dans le cas de mappes de non-entités, lorsque la propriété POJOKeyIndex du HashIndex a la valeur `true`, l'index HashIndex est un HashIndex de clés et c'est la partie clé de l'entrée qui sera utilisée pour l'indexation. Lorsque la propriété AttributeName du HashIndex n'est pas spécifiée, la totalité de la clé est indexée. Sinon, le HashIndex de clés ne peut être qu'un HashIndex à attribut unique.

Par exemple, l'ajout de la propriété suivante dans l'exemple précédent transforme le HashIndex en HashIndex de clés car la propriété POJOKeyIndex a la valeur `true`.

```
<property name="POJOKeyIndex" type="boolean" value="true"
description="indicates if POJO key HashIndex" />
```

Dans l'index clé de l'exemple précédent, la propriété AttributeName ayant `employeeCode` pour valeur, l'attribut indexé est le champ employeeCode de la partie clé des entrées de mappe. Pour générer l'index sur la totalité de la partie clé des entrées de mappe, il faudrait supprimer la propriété AttributeName.

HashIndex de plage

Lorsque la propriété RangeIndex de l'index HashIndex a la valeur `true`, l'index HashIndex est un index de plage et il peut prendre en charge l'interface MapRangeIndex. Une interface MapRangeIndex prend en charge des fonctions de

recherche de données à l'aide des fonctions de plage (range) telles que greater than, less than ou les deux, alors qu'un index MapIndex prend uniquement en charge les fonctions d'égalité (equals). Pour un index à attribut unique, la propriété RangeIndex ne peut avoir la valeur `true` que si l'attribut indexé est de type Comparable. Si l'index à attribut unique est utilisé par la requête, la propriété RangeIndex doit avoir la valeur `true` et l'attribut indexé doit être de type Comparable. Pour l'index HashIndex de relation et l'index HashIndex composite, la propriété RangeIndex doit avoir la valeur `false`.

L'exemple qui précède est un index HashIndex de plage car la propriété RangeIndex a la valeur `true`.

Le tableau qui suit récapitule l'utilisation de l'index de plage.

Tableau 7. Prise en charge de l'index de plage. Indique si les types de HashIndex prennent ou non en charge l'index de plage.

Type de HashIndex	**Prise en charge de l'index de plage**
HashIndex à attribut unique : la clé ou l'attribut indexé est de type Comparable	Oui
HashIndex à attribut unique : la clé ou l'attribut indexé n'est pas de type Comparable	Non
HashIndex composite	Non
HashIndex de relation	Non

Optimisation de requête à l'aide de HashIndex

Des index judicieusement définis et utilisés ne peuvent qu'améliorer considérablement les performances des requêtes. Les requêtes WebSphere eXtreme Scale peuvent utiliser les plug-in HashIndex pré-intégrés pour optimiser leurs performances. Même si l'utilisation des index peut considérablement améliorer les performances des requêtes, elle n'est pas sans impact sur les performances des opérations transactionnelles des mappes.

Configuration d'expulseurs

Les expulseurs peuvent être configurés à l'aide du fichier XML du descripteur d'ObjectGrid ou d'un programme.

Pourquoi et quand exécuter cette tâche

Vous trouverez des informations de référence sur la configuration par XML d'expulseurs dans «Fichier XML du descripteur d'ObjectGrid», à la page 143.

Expulseur TimeToLive (TTL)

WebSphere eXtreme Scale fournit un mécanisme par défaut pour expulser les entrées du cache et un plug-in permettant la création d'expulseurs personnalisés. Un expulseur contrôle l'appartenance des entrées dans chaque instance de BackingMap.

Activation de l'expulseur basé sur la durée de vie à l'aide d'un programme

Les expulseurs basés sur la durée de vie sont associés à des instances de BackingMap. L'expulseur par défaut utilise une stratégie d'expulsion basée sur la

durée de vie pour chaque instance de BackingMap. Si vous fournissez un mécanisme d'expulsion connectable, il utilise généralement une stratégie d'expulsion basée sur le nombre d'entrées au lieu de la durée de vie.

Le fragment de code suivant utilise l'interface BackingMap pour définir un délai d'expiration de 10 minutes pour chaque entrée après sa création.

```
expulseur basé sur la durée de vie à l'aide d'un
programmeimport 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 );
```

L'argument de la méthode setTimeToLive est 600 car cela indique la valeur de durée de vie en secondes. Le code précédent doit être exécuté avant l'appel de la méthode d'initialisation sur l'instance ObjectGrid. Ces attributs BackingMap ne peuvent pas être modifiés une fois que l'instance ObjectGrid est initialisée. Une fois que le code est exécuté, une entrée insérée dans BackingMap myMap possède un délai d'expiration. Une fois ce délai expiré, l'expulseur basé sur la durée de vie supprime l'entrée.

Pour définir un délai d'expiration à la date du dernier accès plus 10 minutes, changez l'argument qui est passé à la méthode setTtlEvictorType de TTLType.CREATION_TIME à TTLType.LAST_ACCESS_TIME. Avec cette valeur, le délai d'expiration équivaut à la date du dernier accès plus 10 minutes. Quand une entrée vient d'être créée, sa date de dernier accès est sa date de création. Pour baser l'heure de l'expiration sur la dernière *modification* au lieu de se contenter du dernier *accès* (qu'une modification ait eu effectivement lieu ou non), remplacez le paramètre TTLType.LAST_ACCESS_TIME par TTLType.LAST_UPDATE_TIME.

Lorsque vous utilisez TTLType.LAST_ACCESS_TIME ou TTLType.LAST_UPDATE_TIME, vous pouvez utiliser les interfaces ObjectMap et JavaMap pour que celles-ci substituent leur propre valeur de durée de vie de la mappe de sauvegarde. Ce mécanisme autorise une application à utiliser une valeur de durée de vie différente pour chaque entrée créée. Partons du principe que le fragment de code précédent définit l'attribut ttlType sur LAST_ACCESS_TIME et la valeur de durée de vie sur 10 minutes. Une application peut substituer sa propre valeur de durée de vie pour chacune des entrées en exécutant le code suivant avant de créer ou de modifier l'entrée :

```
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" );
```

Dans le fragment de code précédent, l'entrée avec la clé key1 a un délai d'expiration basé sur la date d'insertion plus 30 minutes en raison de l'appel de la méthode setTimeToLive(1800) sur l'instance ObjectMap. La variable oldTimeToLive1 est définie sur 600 car la valeur de durée de vie de BackingMap

est utilisée comme variable par défaut si la méthode setTimeToLive n'a pas été appelée précédemment sur l'instance ObjectMap.

L'entrée avec la clé key2 a un délai d'expiration basé sur la date d'insertion plus 20 minutes en raison de l'appel de la méthode setTimeToLive(1200) sur l'instance ObjectMap. La variable oldTimeToLive2 est définie sur 1800 car la valeur de durée de vie de l'appel précédent de la méthode ObjectMap.setTimeToLive définit cette valeur sur 1800.

Les exemples précédents montrent l'insertion de deux entrées de mappe dans la mappe myMap pour les clés key1 et key2. Plus tard, l'application peut toujours mettre à jour ces entrées de mappe tout en conservant les valeurs de durée de vie utilisées au moment de l'insertion pour chaque entrée de mappe. L'exemple suivant illustre comment conserver les valeurs de durée de vie, en utilisant une constante définie dans l'interface 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();
```

Comme la valeur spéciale ObjectMap.USE_DEFAULT est utilisée sur l'appel de la méthode setTimeToLive, la clé key1 conserve sa valeur de 1800 secondes et la clé key2 conserve la sienne de 1200 secondes, car ces valeurs on été utilisées quand les entrées de mappe ont été insérées par la transaction précédente.

L'exemple précédent montre également une nouvelle entrée de mappe pour l'insertion de la clé key3. Dans ce cas, la valeur spéciale USE_DEFAULT indique d'utiliser le paramètre par défaut de valeur de durée de vie pour cette mappe. La valeur par défaut est définie par l'attribut BackingMap de durée de vie. Voir Les attributs de l'interface BackingMap pour en savoir plus sur la manière dont est redéfini l'attribut de durée de vie sur l'instance BackingMap.

Voir la documentation d'API pour la méthode setTimeToLive sur les interfaces ObjectMap et JavaMap. La documentation explique qu'une exception IllegalStateException survient si la méthode BackingMap.getTtlEvictorType renvoie autre chose que la valeur TTLType.LAST_ACCESS_TIME ou TTLType.LAST_UPDATE_TIME. Les interfaces ObjectMap et JavaMap ne peuvent substituer la valeur de durée de vie que lorsqu'on utilise le paramètre LAST_ACCESS_TIME ou TTLType.LAST_UPDATE_TIME pour le type d'expulseur TTL. La méthode setTimeToLive ne peut pas être utilisée pour substituer sa propre valeur de durée de vie lorsqu'on utilise les paramètres CREATION_TIME ou NONE.

Activation de l'expulseur basé sur la durée de vie à l'aide d'une configuration XML

Au lieu d'utiliser l'interface BackingMap pour définir à l'aide d'un programme les attributs BackingMap qui doivent être utilisés par l'expulseur basé sur la durée de vie, vous pouvez utiliser un fichier XML pour configurer chaque instance de BackingMap. Le code suivant démontre comment définir ces attributs pour trois mappes BackingMap différentes :

Activation de l'expulseur basé sur la durée de vie par 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|LAST_UPDATE_TIME"
     timeToLive="1800" />
        <backingMap name="map3" ttlEvictorType="CREATION_TIME"
     timeToLive="1200" />
    </objectgrid>
</objectGrids>
```

L'exemple précédent montre que l'instance BackingMap map1 utilise le type d'expulseur basé sur la durée de vie NONE. L'instance BackingMap map2 utilise un type d'expulseur basé sur la durée de vie LAST_ACCESS_TIME TTL ou TTLType.LAST_UPDATE_TIME (ne spécifiez que l'un ou l'autre de ces deux paramètres) et sa durée de vie est de 1800 secondes(30 minutes). L'instance BackingMap map3 est définie pour utiliser le type d'expulseur basé sur la durée de vie CREATION_TIME et a une durée de vie de 1200 secondes (20 minutes).

Connecter un expulseur

Les expulseurs étant associés à des mappes de sauvegarde, l'interface BackingMap permet de spécifier l'expulseur à connecter.

Expulseurs optionnels

L'expulseur TTL par défaut utilise une stratégie d'expulsion basée sur le temps, et le nombre d'entrées dans la mappe de sauvegarde n'a pas d'effet sur le délai d'expiration d'une entrée. Au lieu de vous baser sur le temps, vous pouvez connecter un expulseur optionnel pour expulser des entrées en fonction du nombre d'entrées existantes.

Les expulseurs optionnels fournissent des algorithmes communément utilisés pour décider quelles entrées expulser dès qu'une mappe de sauvegarde dépasse une certaine taille.

- L'expulseur LRUEvictor utilise un algorithme déterminant les entrées les moins récemment utilisées (LRU).
- L'expulseur LFUEvictor utilise un algorithme déterminant les entrées les moins fréquemment utilisées(LFU).

La mappe de sauvegarde informe un expulseur quand des entrées sont créées, modifiées ou supprimées d'une transaction. La mappe de sauvegarde suit ces entrées et choisit quand expulser de l'instance BackingMap une ou plusieurs de ces entrées.

Une instance BackingMap ne dispose pas d'informations de configuration pour une taille maximale. A la place, les propriétés d'expulseur sont définies pour contrôler le comportement de ce dernier. Le LRUEvictor et le LFUEvictor ont une taille maximale utilisée pour déclencher l'expulsion des entrées quand une certaine taille est dépassée. Comme l'expulseur TTL, il est possible que les expulseurs LRU et LFU n'expulsent pas immédiatement une entrée quand le nombre maximal d'entrées est atteint, pour minimiser l'impact sur les performances.

Si l'algorithme d'expulsion LRU ou LFU se révèle inadéquat pour une application particulière, vous pouvez écrire vos propres expulseurs pour créer votre stratégie d'expulsion.

Connecter des expulseurs optionnels

Pour ajouter des expulseurs optionnels à la configuration BackingMap, vous pouvez utiliser une configuration par programmation ou une configuration XML.

Connecter un expulseur par programmation

Les expulseurs étant associés à des mappes de sauvegarde, l'interface BackingMap permet de spécifier l'expulseur à connecter. Le fragment de code qui suit montre comment spécifier un expulseur LRUEvictor pour la mappe de sauvegarde map1 et un expulseur LFUEvictor pour l'instance BackingMap map2 :

```
connecter un expulseur par programmation
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);
```

Le fragment de code montrait un expulseur LRUEvictor utilisé pour la mappe de sauvegarde map1 avec un nombre maximum d'entrées d'environ 53 000 (53 * 1000). L'expulseur LFUEvictor était utilisé pour la mappe de sauvegarde map2 avec un nombre maximum d'entrées d'à peu près 422 000 (211*2000). Les deux expulseurs LRU et LFU comportent une propriété SleepTime qui indique au bout de combien de temps l'expulseur doit s'éveiller pour vérifier s'il n'y a pas lieu d'expulser des entrées. Cette durée d'inactivité est spécifiée en secondes. 15 secondes est un bon compromis entre l'impact sur les performances et le souci d'empêcher la mappe de trop grossir. L'objectif est d'utiliser le délai d'inactivité le plus long possible sans que la mappe atteigne une taille excessive.

La méthode setNumberOfLRUQueues définit la propriété LRUEvictor qui indique combien de files d'attente LRU l'expulseur utilise pour gérer les informations LRU. Un ensemble de files d'attente est en effet utilisé pour éviter que chaque entrée ne conserve les informations LRU dans la même file. Cette approche permet d'améliorer les performances en réduisant le nombre d'entrées de mappe qui ont besoin de synchronisation dans le même objet queue. Augmenter le nombre de files d'attente est un bon moyen de réduire l'impact possible de l'expulseur LRU sur les performances. 10 % du nombre maximum d'entrées est un bon point de départ pour définir le nombre des files d'attente. En règle générale, il vaut mieux utiliser un nombre premier. La méthode setMaxSize indique combien d'entrées sont autorisées dans chaque file d'attente. Lorsqu'une file atteint son nombre maximum

d'entrées, la ou les entrées les moins récemment utilisées dans cette file sont expulsées lors de la prochaine vérification par l'expulseur.

La méthode setNumberOfHeaps définit la propriété LFUEvictor qui indique combien d'objets binaires de segments de mémoire LFUEvictor utilise pour gérer les informations LFU. Ici aussi, un ensemble est utilisé afin d'améliorer les performances. Et ici aussi, 10 % du nombre d'entrées et un bon point de départ et il vaut mieux utiliser un nombre premier. La méthode setMaxSize indique combien d'entrées sont autorisées dans chaque segment de mémoire. Lorsqu'un segment de mémoire atteint son nombre maximum d'entrées, la ou les entrées les moins fréquemment utilisées dans ce segment sont expulsées lors de la prochaine vérification par l'expulseur.

Connecter un expulseur par configuration XML

Au lieu d'utiliser diverses API pour programmer la connexion d'un expulseur et pour définir ses propriétés, il est possible d'utiliser un fichier XML pour configurer chacune des mappes de sauvegarde :

connecter un expulseur par 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>
```

Expulsion basée sur la mémoire

Tous les expulseurs pré-intégrés prennent en charge l'expulsion basée sur la mémoire, laquelle peut être activée dans l'interface BackingMap en donnant à l'attribut evictionTriggers la valeur MEMORY_USAGE_THRESHOLD. Pour plus d'informations sur la configuration de l'attribut evictionTriggers dans BackingMap, voir les informations de référence concernant l'interface BackingMap et la configuration d'eXtreme Scale.

L'expulsion basée sur la mémoire repose sur un seuil d'utilisation d'un segment de mémoire. Quand cette expulsion est activée dans la mappe de sauvegarde et que cette dernière dispose d'un expulseur pré-intégré, le seuil d'utilisation est défini en un pourcentage par défaut de la mémoire totale, si le seuil n'a pas été défini auparavant.

Pour modifier le pourcentage du seuil d'utilisation, définissez la propriété memoryThresholdPercentage dans les fichiers de propriétés de conteneur et de serveur pour les processus de serveur eXtreme Scale. Pour définir le seuil

d'utilisation cible dans un processus de client d'eXtreme Scale, vous pouvez utiliser le MemoryPoolMXBean. Voir aussi : le fichier containerServer.props et Démarrage de processus serveur eXtreme Scale.

Si, pendant l'exécution, l'utilisation de la mémoire dépasse le seuil cible, les expulseurs basés sur la mémoire commencent à expulser des entrées et essaient de conserver l'utilisation de la mémoire en dessous du seuil cible. Cependant, il n'existe aucune garantie que la vitesse d'expulsion soit assez grande pour éviter une erreur de dépassement de mémoire si l'exécution du système continue à rapidement consommer de la mémoire.

Configuration d'une stratégie de verrouillage

Vous pouvez définir une stratégie optimiste, pessimiste ou sans verrouillage sur chaque BackingMap de la configuration de WebSphere eXtreme Scale.

Pourquoi et quand exécuter cette tâche

Vous pouvez spécifier une stratégie de verrouillage à l'aide d'un programme ou d'un fichier XML. Pour plus d'informations sur le verrouillage, voir les informations sur les stratégies de verrouillage dans la *Présentation du produit*.

Procédure

- **Configurez une stratégie de verrouillage optimiste**
  - A l'aide d'un programme

    specify optimistic strategy programmatically

    ```
    import com.ibm.websphere.objectgrid.BackingMap;
    import com.ibm.websphere.objectgrid.LockStrategy;
    import com.ibm.websphere.objectgrid.ObjectGrid;
    import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
    ...
    ObjectGrid og =
     ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test");
    BackingMap bm = og.defineMap("optimisticMap");
    bm.setLockStrategy( LockStrategy.OPTIMISTIC );
    ```

  - Utilisation de XML

    specify optimistic strategy using 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>
    ```

- **Configurez une stratégie de verrouillage pessimiste**
  - A l'aide d'un programme

    specify pessimistic strategy programmatically

    ```
    import com.ibm.websphere.objectgrid.BackingMap;
    import com.ibm.websphere.objectgrid.LockStrategy;
    import com.ibm.websphere.objectgrid.ObjectGrid;
    import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
    ...
    ObjectGrid og =
     ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test");
    BackingMap bm = og.defineMap("pessimisticMap");
    bm.setLockStrategy( LockStrategy.PESSIMISTIC);
    ```

– Utilisation de XML

```
specify pessimistic strategy using 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>
```

- **Configurez une stratégie sans verrouillage**
  – A l'aide d'un programme

```
specify a no-locking strategy programmatically

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);
```

  – Utilisation de XML

```
specify a no-locking strategy with 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>
```

Que faire ensuite

Pour éviter une exception java.lang.IllegalStateException, vous devez appeler la méthode setLockStrategy avant d'appeler les méthodes initialize ou getSession sur l'instance ObjectGrid.

Configuration de chargeurs

L'implémentation d'un chargeur implique la configuration de plusieurs attributs.

Remarques sur le préchargement

Les chargeurs sont des plug-in de mappe de sauvegarde appelés lorsque des modifications sont apportées à la mappe de sauvegarde ou lorsque cette dernière est dans l'impossibilité de répondre à une demande de données (absence dans le cache). Pour une présentation de l'interaction d'eXtreme Scale avec un chargeur, consultez les informations relatives aux scénarios de mise en cache en ligne dans la *Présentation du produit*.

Chaque mappe de sauvegarde est associée à un attribut booléen preloadMode défini pour indiquer si le préchargement d'une mappe s'exécute de façon asynchrone. Par défaut, l'attribut preloadMode est défini sur false, ce qui signifie que l'initialisation de la mappe de sauvegarde ne s'effectue que si le préchargement de la mappe est terminé. Par exemple, l'initialisation de la mappe de sauvegarde ne s'effectue que si la méthode preloadMap est renvoyée. Si la méthode preloadMap reconnaît une grande quantité de données en provenance de son programme d'arrière plan et les charge dans la mappe, son exécution peut prendre un certain temps. Dans ce cas, vous pouvez configurer l'utilisation du préchargement asynchrone pour une mappe de sauvegarde donnée en définissant l'attribut preloadMode sur true. Ce paramètre permet au code d'initialisation de la mappe de sauvegarde de démarrer une unité d'exécution qui appelle la méthode preloadMap, permettant ainsi l'initialisation d'une mappe de sauvegarde parallèlement au chargement de la mappe.

Dans un scénario eXtreme Scale réparti, l'un des motifs de préchargement est le préchargement client. Dans le motif de préchargement client, un client eXtreme Scale est responsable de l'extraction des données à partir du programme d'arrière plan et de l'insertion des données dans le serveur eXtreme Scale à l'aide d'agents DataGrid. En outre, le préchargement client peut être exécuté dans la méthode Loader.preloadMap dans une seule et unique partition spécifique. Dans ce cas, le chargement asynchrone des données dans la grille devient crucial. Si le préchargement client était exécuté dans la même unité d'exécution, la mappe de sauvegarde ne serait jamais initialisée, de sorte que la partition dans laquelle elle réside ne passerait jamais au statut ONLINE. Par conséquent, le client eXtreme Scale ne pourrait pas envoyer la demande à la partition, ce qui provoquerait une exception au final.

Si un client eXtreme Scale est utilisé dans la méthode preloadMap, il est recommandé de définir l'attribut preloadMode sur true. Il est également possible de lancer une unité d'exécution dans le code de préchargement du client.

Le fragment de code ci-dessous illustre comment l'attribut preloadMode est défini pour activer le préchargement asynchrone :

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

L'attribut preloadMode peut également être défini à l'aide d'un fichier XML tel qu'illustré dans l'exemple suivant :

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

Objet TxID et utilisation de l'interface TransactionCallback

Les méthodes get et batchUpdate dans l'interface Loader reçoivent un objet TxID qui représente la transaction Session nécessitant l'exécution de l'opération get ou batchUpdate. Il est possible d'appeler les méthodes get et batchUpdate plusieurs fois pour une même transaction. Par conséquent, les objets inclus dans l'étendue de la transaction nécessaires au chargeur sont généralement conservés dans un emplacement de l'objet TxID. Un chargeur JDBC (Java Database Connectivity) est utilisé pour illustrer comment un chargeur utilise l'objet TxID et les interfaces TransactionCallback.

Il est également possible de stocker plusieurs mappes ObjectGrid dans la même base de données. Chaque mappe possède son propre chargeur et il se peut que chaque chargeur doive se connecter à la même base de données. Lors de la

connexion à la même base de données, chaque chargeur vise à utiliser la même connexion JDBC de façon à valider les modifications apportées à chaque table dans le cadre de la même transaction de base de données. En général, l'écriture de l'implémentation du chargeur est effectuée par la même personne qui écrit l'implémentation de l'interface TransactionCallback. Lors de l'extension de l'interface TransactionCallback, la meilleure pratique consiste à ajouter des méthodes requises par le chargeur pour établir la connexion à la base de données et pour mettre en cache les instructions préparées. Cette méthode devient une évidence lorsque vous constatez comment les interfaces TransactionCallback et TxID sont utilisées par le chargeur.

Par exemple, le chargeur peut nécessiter l'extension de l'interfaceTransactionCallback de la manière suivante :

```
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 );
}
```

Ces nouvelles pratiques peuvent permettre aux méthodes get et batchUpdate du chargeur d'établir une connexion comme suit :

```
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;
}
```

Dans l'exemple précédent et dans les exemples qui suivent, ivTcb et ivOcb sont des variables d'instance du chargeur qui ont été initialisées tel que décrit dans la section Remarques sur le préchargement. La variable ivTcb est une référence à l'instance MyTransactionCallback et la variable ivOcb est une référence à l'instance MyOptimisticCallback. La variable databaseName est une variable d'instance du chargeur qui a été définie en tant que propriété du chargeur lors de l'initialisation de la mappe de sauvegarde. L'argument isolationLevel est l'une des constantes de la connexion JDBC définies pour les différents niveaux d'isolement pris en charge par JDBC. Si le chargeur utilise une implémentation optimiste, la méthode get utilise généralement une connexion JDBC de validation automatique pour extraire les données de la base de données. Dans ce cas, il se peut que le chargeur possède une méthode getAutoCommitConnection implémentée de la façon suivante :

```
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;
}
```

Rappelez-vous que la méthode batchUpdate comporte l'instruction switch suivante :

```
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;
}
```

Chacun méthode buildBatchSQL utilise l'interface MyTransactionCallback pour obtenir une instruction préparée. Le fragment de code ci-dessous présente la méthode buildBatchSQLUpdate créant une instruction SQL update pour mettre à jour une entrée EmployeeRecord et l'ajouter pour la mise à jour par lots :

```
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();
}
```

Une fois que la boucle batchUpdate a créé toutes les instructions préparées, elle appelle la méthode getPreparedStatementCollection. Cette dernière est implémentée comme suit :

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

Lorsque l'application appelle la méthode commit sur la session, le code de session appelle la méthode commit sur la méthode TransactionCallback après avoir envoyé toutes les modifications apportées par la transaction vers le chargeur pour chaque mappe qui a été modifiée par la transaction. Etant donné que tous les chargeurs ont utilisé la méthode MyTransactionCallback pour établir la connexion et les instructions préparées requises, la méthode TransactionCallback identifie la connexion à utiliser pour demander la validation des modifications par la programme d'arrière plan. L'extension de l'interface TransactionCallback avec des méthodes requises par chaque chargeur présente donc les avantages suivants :

- L'objet TransactionCallback incorpore l'utilisation des emplacements TxID pour les données incluses dans l'étendue de la transaction et le chargeur n'a pas besoin des informations sur les emplacements TxID. Le chargeur doit uniquement reconnaître les méthodes ajoutées à l'objet TransactionCallback utilisant l'interface MyTransactionCallback pour les fonctions de prise en charge requises par le chargeur.

- L'objet TransactionCallback peut permettre le partage de la connexion entre chaque chargeur qui établit la connexion avec le même programme d'arrière plan pour éviter un protocole commit à deux phases.
- L'objet TransactionCallback peut faire en sorte que la connexion au programme d'arrière plan est terminée via une instruction commit ou rollback appelée lors de la connexion si nécessaire.
- TransactionCallback garantit le nettoyage des ressources de la base de données à la fin d'une transaction.
- TransactionCallback se masque s'il établit une connexion gérée à partir d'un environnement géré tel que WebSphere Application Server ou autre serveur d'applications compatible Java 2 Platform, Enterprise Edition (J2EE). Cet avantage permet l'utilisation du même code de chargeur dans un environnement géré et dans un environnement non géré. Seul le plug-in TransactionCallback doit être modifié.
- Pour plus de détails sur l'utilisation des emplacements TxID par l'implémentation TransactionCallback pour les données incluses dans l'étendue de la transaction, reportez-vous à la rubrique Plug-in TransactionCallback.

OptimisticCallback

Comme mentionné précédemment, le chargeur peut utiliser une approche optimiste pour le contrôle des accès simultanés. Dans ce cas, l'exemple de la méthode buildBatchSQLUpdate doit être légèrement modifié pour l'implémentation d'une approche optimiste. Il existe différentes façons pour mettre en oeuvre une approche optimiste. La façon la plus courante consiste à disposer d'une colonne d'horodatage ou d'une colonne de compteur de numéros de séquence pour la gestion des versions de mise à jour de la ligne. Supposons que la table employee comporte une colonne de numéros de séquence qui s'incrémentent à chaque mise à jour de la ligne. Vous modifiez ensuite la signature de la méthode buildBatchSQLUpdate pour qu'elle soit transmise à l'objet LogElement et non à la paire clé-valeur. Elle doit également utiliser l'objet OptimisticCallback relié à la mappe de sauvegarde pour l'obtention de l'objet de version initiale et la mise à jour de l'objet de version. L'exemple suivant est celui d'une méthode buildBatchSQLUpdate modifiée qui utilise la variable d'instance ivOcb initialisée tel que décrit dans la section preloadMap :

modified batch-update method code example

```
private void buildBatchSQLUpdate( TxID tx, LogElement le, Connection conn )
 throws SQLException, LoaderException
{
    // Obtenir l'objet de version initiale lors de la dernière lecture ou
    // mise à jour de cette entrée
    // de mappe dans la base de données.
    Employee emp = (Employee) le.getCurrentValue();
    long initialVersion = ((Long) le.getVersionedValue()).longValue();
    // Obtenir l'objet de version de l'objet Employee mis à jour pour
    // l'opération SQL update.
    Long currentVersion = (Long)ivOcb.getVersionedObjectForValue( emp );
    long nextVersion = currentVersion.longValue();
    // Créer maintenant l'instruction SQL update qui inclut l'objet
    // de version dans la clause where
    // pour la vérification optimiste.
    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();
}
```

L'exemple montre que l'objet LogElement est utilisé pour obtenir la valeur de la version initiale. Lorsque la transaction accède pour la première fois à l'entrée de mappe, un objet LogElement est créé avec l'objet Employee initial obtenu à partir de la mappe. L'objet Employee initial est également transmis à la méthode getVersionedObjectForValue dans l'interface OptimisticCallback et le résultat est enregistré dans l'objet LogElement. Ce traitement a lieu avant qu'une application soit référencée à l'objet Employee initial et qu'elle appelle une méthode modifiant l'état de l'objet Employee initial.

L'exemple montre que le chargeur utilise la méthode getVersiondObjectForValue pour obtenir l'objet de version pour l'objet Employee mis à jour. Avant d'appeler la méthode batchUpdate dans l'interface Loader, eXtreme Scale appelle la méthode updateVersionedObjectForValue dans l'interface OptimisticCallback pour provoquer la génération d'un nouvel objet de version pour l'objet Employee mis à jour. Une fois que la méthode batchUpdate est renvoyée à l'ObjectGrid, l'objet LogElement est mis à jour avec l'objet de version actuelle et devient le nouvel objet de version initiale. Cette étape est nécessaire parce qu'il se peut que l'application ait appelé la méthode flush sur la mappe et non la méthode commit sur la session. Il est possible que le chargeur soit appelé plusieurs fois par une seule transaction pour la même clé. C'est pourquoi, eXtreme Scale fait en sorte que l'objet LogElement soit mis à jour avec l'objet de nouvelle version à chaque mise à jour de la ligne dans la table employee.

Désormais que le chargeur détient l'objet de version initiale et l'objet de prochaine version, il peut exécuter une instruction SQL update qui définit la colonne SEQNO sur la valeur de l'objet de prochaine version et utilise cette dernière dans la clause where. Cette approche est parfois désignée comme étant une instruction update sur-qualifiée. L'utilisation d'une instruction update sur-qualifiée permet à la base de données relationnelle de vérifier que la ligne n'a pas été modifiée par une autre transaction entre la lecture des données de la base de données par cette transaction et la mise à jour de la base de données par cette transaction. Si une autre transaction a modifié la ligne, le tableau de comptage renvoyé par la mise à jour par lots indique qu'aucune ligne n'a été mise à jour pour cette clé. Le chargeur est chargé de vérifier que l'opération SQL update n'a pas mis à jour la ligne. Si c'est le cas, il affiche une exception com.ibm.websphere.objectgrid.plugins.OptimisticCollisionException pour informer la session que la méthode batchUpdate a échoué en raison de la tentative de plusieurs transactions simultanées de mettre à jour la même ligne dans la table de base de données. Cette exception provoque l'annulation de la session et l'application doit relancer la transaction entière. La justification réside dans le fait que la nouvelle tentative aboutira, ce qui explique pourquoi cette approche est appelée optimiste. L'approche optimiste est plus performante si les données sont peu modifiées ou si les transactions simultanées tentent rarement de mettre à jour la même ligne.

Le chargeur doit impérativement utiliser le paramètre de clé du constructeur OptimisticCollisionException pour identifier quelle clé ou quel ensemble de clés a provoqué l'échec de la méthode optimiste batchUpdate. Le paramètre de clé peut être l'objet clé proprement dit ou un tableau d'objets clé si plusieurs clés ont abouti à l'échec de la mise à jour optimiste. eXtreme Scale utilise la méthode getKey du constructeur OptimisticCollisionException pour déterminer quelles entrées de

mappe contiennent les données périmées et ont provoqué l'exception. Un aspect du processus d'annulation consiste à expulser les entrées de mappe périmées de la mappe. Cette expulsion s'avère nécessaire pour que toutes les transactions suivantes accédant aux mêmes clés déclenchent la méthode get de l'interface Loader appelée pour actualiser les entrées de mappe avec les données actuelles de la base de données.

Autres modes d'implémentation d'une approche optimiste par un chargeur :

- Il n'existe aucune colonne d'horodatage ou de numéro de séquence. Dans ce cas, la méthode getVersionObjectForValue de l'interface OptimisticCallback renvoie simplement l'objet de valeur en tant que version. Avec cette approche, le chargeur doit créer une clause WHERE comprenant tous les champs l'objet version initial. Cette approche n'est pas efficace et tous les types de colonnes ne sont pas admissibles pour une utilisation dans la clause WHERE d'une instruction SQL update sur-qualifiée. Cette approche n'est généralement pas utilisée.
- Il n'existe aucune colonne d'horodatage ou de numéro de séquence. Cependant, contrairement à l'approche précédente, la clause WHERE contient uniquement les champs de valeur modifiées par la transaction. L'une des méthodes permettant de détecter les champs qui ont été modifiés consiste à donner au mode de copie de la mappe de sauvegarde la valeur CopyMode.COPY_ON_WRITE. Ce mode de copie exige la transmission d'une interface de valeur à la méthode setCopyMode dans l'interface de la mappe de sauvegarde. La mappe de sauvegarde crée des objets proxy dynamiques qui implémentent l'interface de valeur fournie. Avec ce mode de copie, le chargeur peut transtyper chaque valeur en objet com.ibm.websphere.objectgrid.plugins.ValueProxyInfo. L'interface ValueProxyInfo comporte une méthode qui permet au chargeur d'obtenir la liste des noms d'attributs modifiés par la transaction. Cette méthode permet au chargeur d'appeler les méthodes get dans l'interface de valeur pour les noms d'attributs afin de consulter les données modifiées et créer une instruction SQL update qui définit uniquement les attributs modifiés. La clause where peut désormais être créée en incluent la colonne de clé primaire plus chacune des colonnes d'attributs modifiés. Cette approche est plus efficace que la précédente mais elle exige l'écriture de plus de code dans le chargeur et un cache de l'instruction préparée potentiellement plus important pour le traitement des différentes permutations. Toutefois, si les transactions modifient généralement que quelques attributs, cette limitation peut ne pas être problématique.
- Certaines bases de données relationnelle peuvent comporter une API pour la maintenance automatique des données de colonne particulièrement utiles pour la gestion optimiste des versions. Pour plus d'informations sur l'existence de ces possibilités, consultez la documentation de votre base de données.

Configuration de la prise en charge d'un loader en écriture différée

Vous pouvez activer l'écriture différée en utilisant le fichier XML de descripteur d'ObjectGrid ou par programmation avec l'interface BackingMap.

Pour activer l'écriture différée, vous pouvez utiliser le fichier XML de descripteur d'ObjectGrid ou passer pa la programmation avec l'interface BackingMap.

Fichier XML de descripteur d'ObjectGrid

Lors de la configuration d'un ObjectGrid à l'aide d'un fichier XML de descripteur d'ObjectGrid, le chargeur en écriture différée est activé en définissant l'attribut write-behind dans la balise backingMap. Voici un exemple :

```
<objectGrid name="library" >
     <backingMap name="book" writeBehind="T300;C900" pluginCollectionRef="bookPlugins"/>
```

Dans l'exemple ci-dessus, l'écriture différée par la mappe de sauvegarde book est activée avec le paramètre T300;C900. L'attribut write-behind spécifie le temps de mise à jour maximal et/ou le nombre de mises à jour de clés maximal. Le format du paramètre write-behind est le suivant :

```
::= <defaults> | <update time> | <update key count> | <update time> ";"
<update key count>::= "T" <positive integer>::= "C" <positive integer>::= ""
```

- attribut write-behind d'écriture différée
- temps de mise à jour
- nombre de clés à mettre à jour
- valeurs par défaut

Les mises à jour du chargeur ont lieu lorsque l'un des événements suivants se produit :

1. Le temps de mise à jour maximal, en secondes, s'est écoulé depuis la dernière mise à jour.
2. Le nombre de clés mises à jour dans la mappe de files d'attente a atteint le nombre de clés à mettre à jour.

Ces paramètres sont uniquement indicatifs. Le temps et le nombre réels de mises à jour seront compris dans une plage de paramètres proche. Toutefois, nous ne garantissons pas que le temps et le nombre réels de mises à jour soient identiques à ceux définis dans les paramètres. De plus, la première mise à jour suivante est susceptible d'avoir lieu dans un délai supérieur au double de celui de la mise à jour. Cela est dû au fait qu'ObjectGrid répartit au hasard l'heure de début des mises à jour de manière à ce que toutes les partitions n'accèdent pas à la base de données en même temps.

Dans l'exemple précédent T300;C900, le chargeur écrit les données dans le système dorsal lorsque 300 secondes se sont écoulées depuis la dernière mise à jour ou lorsque 900 clés sont en attente de mise à jour. Par défaut, le temps de mise à jour est de 300 secondes et le nombre de clés à mettre à jour est 1 000.

Mise en cache à écriture différée

Vous pouvez utiliser la mise en cache à écriture différée pour réduire le temps système supplémentaire nécessaire lors de la mise à jour d'une base de données utilisée en tant que base de données dorsale.

Introduction

La mise en cache à écriture différée met les mises à jour en file d'attente de manière asynchrone dans le plug-in du Loader. Une amélioration des performances est possible si vous déconnectez les mises à jour, les insertions et les suppressions pour une mappe, le temps système supplémentaire de mise à jour de la base de données dorsale. La mise à jour asynchrone est exécutée après un délai exprimé par une durée (par exemple cinq minutes) ou un délai exprimé par un nombre d'entrées (1000 entrées).

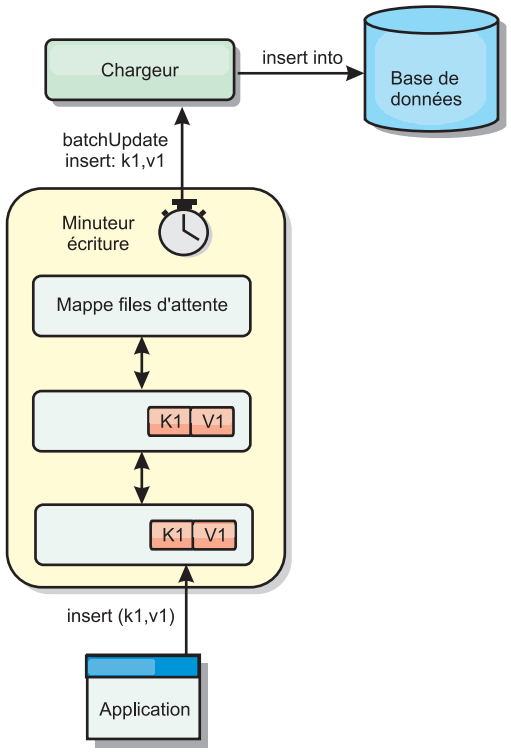


Figure 7. Mise en cache en écriture différée

La configuration à écriture différée sur une mappe de sauvegarde crée une unité d'exécution entre le Loader et la mappe. Le Loader délègue alors les demandes de données via l'unité d'exécution en fonction des paramètres de configuration de la méthode BackingMap.setWriteBehind. Lorsqu'une transaction eXtreme Scale insère, met à jour ou supprime une entrée dans une mappe, un objet LogElement est créé pour chacun de ces enregistrements. Ces éléments sont envoyés au Loader à écriture différée et mis en file d'attente dans une ObjectMap spéciale appelée mappe de files d'attente. Chaque mappe de sauvegarde pour laquelle le paramètre d'écriture différée est activé a ses propres mappes de files d'attente. L'unité d'exécution à écriture différée supprime périodiquement les données mises en file d'attente des mappes correspondantes et les insère dans le Loader dorsal.

Le chargeur à écriture différée envoie uniquement les types insertion, mise à jour et suppression des objets LogElement au chargeur réel. Tous les autres types, par exemple le type EVICT, sont ignorés.

Avantages

L'activation de la prise en charge de l'écriture différée présente les avantages suivants :

- **Isolement en cas d'arrêt anormal de la base de données dorsale :** la mise en cache à écriture différée propose une couche d'isolement en cas d'arrêt anormal de la base de données dorsale. Les mises à jour sont alors placées dans la mappe de files d'attente. Les applications peuvent continuer à envoyer des transactions vers eXtreme Scale. Lors de la reprise du système dorsal, les données contenues dans la mappe de files d'attente sont insérées dans celui-ci.
- **Réduction de la charge du système dorsal :** le chargeur à écriture différée fusionne les mises à jour en fonction des clés de façon qu'une seule mise à jour

fusionnée par clé existe dans la mappe de files d'attente. Cette fusion diminue le nombre de mises à jour dans la base de données dorsale.

- **Amélioration des performances de la transaction :** la durée de chaque transaction eXtreme Scale est réduite car la transaction n'a plus à attendre que les données soient synchronisées avec le système dorsal.

Considérations liées à la conception d'applications

L'activation de la prise en charge de l'écriture différée est simple, mais la conception d'une application devant prendre en charge ce type d'écriture demande réflexion. Sans écriture différée, la transaction ObjectGrid encadre la transaction dorsale. La transaction ObjectGrid démarre avant la transaction dorsale et se termine après celle-ci.

Lorsque la prise en charge de l'écriture différée est activée, la transaction ObjectGrid se termine avant le début de la transaction dorsale. La transaction ObjectGrid et la transaction dorsale sont dissociées.

Contraintes d'intégrité référentielle

Chaque mappe de sauvegarde configurée avec la prise en charge de l'écriture différée possède sa propre unité d'exécution à écriture différée permettant d'insérer les données dans le système dorsal. Les données mises à jour dans les différentes mappes d'une transaction ObjectGrid sont mises à jour dans le système dorsal via différentes transactions dorsales. Par exemple, la transaction T1 met à jour la clé clé1 dans la mappe Mappe1 et la clé clé2 dans la mappe Mappe2. La clé key1 mise à jour dans la mappe Map1 est actualisée vers le système dorsal dans une transaction expéditrice et la clé key2 actualisée dans la mappe Map2 l'est dans une autre transaction expéditrice ; cette mise à jour vers le dorsal est effectuée dans deux unités d'exécution différentes, toutes deux en écriture différée. Si les données stockées dans Map1 et Map2 ont des liens, tels que des contraintes de clé externe dans le système dorsal, les mises à jour sont susceptibles d'échouer.

Lors de la conception de contraintes d'intégrité référentielle dans votre base de données dorsale, vérifiez que les mises à jour désordonnées sont autorisées.

Comportement de la mappe de files d'attente en matière de verrouillage

Le comportement des transactions en matière de verrouillage constitue une autre différence notable. La grille d'objets prend en charge trois stratégies de verrouillage différentes : PESSIMISTIC, OPTIMISITIC et NONE. Les mappes de files d'attente à écriture différée utilisent la stratégie de verrouillage pessimiste, quelle que soit la stratégie de verrouillage configurée pour leur mappe de sauvegarde. Deux types différents d'opérations permettant d'acquérir un verrou sur la mappe de files d'attente existent :

- Lorsqu'une transaction ObjectGrid est validée ou qu'un vidage (de mappe ou de session) se produit, la transaction lit la clé de la mappe de files d'attente et place un verrou S sur la clé.
- Lorsqu'une transaction ObjectGrid est validée, elle tente de mettre à niveau le verrou S vers un verrou X sur la clé.

Du fait de ce comportement supplémentaire, vous pouvez constater des différences dans les comportements relatifs au verrouillage.

- Si la mappe des utilisateurs est configurée en tant que stratégie de verrouillage de type pessimiste, la différence de comportement est peu importante. A chaque appel d'un vidage ou d'une validation, un verrou S est placé sur la même clé de la mappe de files d'attente. Pendant la durée de la validation, un verrou X est acquis pour la clé de la mappe des utilisateurs et pour la clé de la mappe de files d'attente.
- Si la mappe des utilisateurs est configurée en tant que stratégie de verrouillage de type OPTIMISTIC ou NONE, la transaction utilisateur suit le modèle de la stratégie de verrouillage de type PESSIMISTIC. A chaque appel d'un vidage ou d'une validation, un verrou S est acquis pour la même clé de la mappe de files d'attente. Pendant la durée de la validation, un verrou X est acquis pour la clé de la mappe de files d'attente utilisant la même transaction.

Nouvelles tentatives des transactions du chargeur

ObjectGrid ne prend pas en charge les transactions à 2 phases ou transactions XA. L'unité d'exécution à écriture différée supprime les enregistrements de la mappe de files d'attente et met à jour les enregistrements dans le système dorsal. En cas d'échec du serveur au milieu de la transaction, certaines mises à jour dorsales risquent d'être perdues.

Le chargeur à écriture différée tente automatiquement d'écrire à nouveau les transactions ayant échoué et envoie une LogSequence en attente de validation au système dorsal pour éviter toute perte de données. Pour que l'exécution de cette action soit possible, le chargeur doit être idempotent, ce qui signifie que lorsque Loader.batchUpdate(TxId, LogSequence) est appelé deux fois avec la même valeur, le résultat est le même que s'il était appelé une fois. Les implémentations du chargeur doivent implémenter l'interface RetryableLoader pour activer cette fonction. Pour plus de détails, consultez la documentation relative à l'API.

Echec du chargeur

Un échec du plug-in du chargeur est possible lorsque celui-ci ne parvient pas à communiquer avec la base de données dorsale. Cette situation peut se produire si la connexion réseau ou le serveur de base de données est inactif. Le chargeur à écriture différée met les mises à jour en file d'attente et tente périodiquement d'insérer les modifications apportées aux données dans le chargeur. Ce dernier doit signaler le problème de connectivité à l'environnement d'exécution ObjectGrid en générant une exception LoaderNotAvailableException.

L'implémentation du chargeur doit donc pouvoir distinguer un échec lié aux données d'une défaillance physique du chargeur. En cas d'échec lié aux données, une exception LoaderException ou OptimisticCollisionException doit être générée, alors qu'en cas de défaillance physique du chargeur, une exception LoaderNotAvailableException doit être générée. ObjectGrid gère ces deux exceptions de manière différente :

- Si une exception LoaderException est détectée par le chargeur à écriture différée, celui-ci considère que l'échec est lié aux données, par exemple si une clé en double a été identifiée. Le chargeur à écriture différée détaille la mise à jour et examine chaque enregistrement séparément pour isoler la raison de l'échec. Si une exception {{LoaderException}} est à nouveau détectée lors de la mise à jour de l'enregistrement concerné, un enregistrement d'échec de la mise à jour est créé et consigné dans la mappe des mises à jour ayant échoué.
- Si une exception LoaderNotAvailableException est interceptée par le chargeur en écriture différée, celui-ci considère que l'échec est dû à l'impossibilité de se

connecter à la base de données, par exemple, lorsque la base de données dorsale est inactive, lorsque la connexion à une base de données est indisponible ou lorsque le réseau est inactif. Le chargeur en écriture différée attend 15 secondes, puis tente à nouveau la mise à jour par lots de la base de données.

Une erreur fréquente consiste à générer une exception LoaderException alors qu'une exception LoaderNotAvailableException serait plus appropriée. Tous les enregistrements du chargeur à écriture différée deviennent alors des enregistrements d'échec de la mise à jour, ce qui réduit à néant l'objectif de l'isolement en cas d'arrêt anormal du système dorsal.

Considérations sur les performances

La prise en charge de la mise en cache à écriture différée augmente le temps de réponse en supprimant la mise à jour du chargeur de la transaction. Elle permet aussi d'améliorer le débit de la base de données en combinant plusieurs bases de données. Il est important de comprendre le temps système supplémentaire généré par l'unité d'exécution à écriture différée, qui permet de retirer les données de la mappe de files d'attente et de les insérer dans le chargeur.

Le nombre maximal de mises à jour ou leur durée maximale doivent être ajustés en fonction des modèles d'utilisation et de l'environnement attendus. Si la valeur associée à ce nombre ou à cette durée est trop faible, le temps système supplémentaire nécessaire à l'unité d'exécution risque d'être supérieur aux avantages. Le choix d'une valeur élevée pour ces deux paramètres permet d'améliorer l'utilisation de la mémoire lors de la mise en file d'attente des données et retarder le moment de péremption des enregistrements de la base de données.

Pour des performances optimales, réglez les paramètres de l'écriture différée selon les facteurs suivants :

- Rapport entre les transactions de lecture et d'écriture
- Fréquence identique de mise à jour des enregistrements
- Temps d'attente de la mise à jour de la base de données

Prise en charge de la mise en cache en écriture différée

Vous pouvez utiliser la mise en cache en écriture différée pour réduire le temps système occasionné par les mises à jour de la base de données dorsale. La mise en cache en écriture différée met en file d'attente les mises à jour du plug-in Loader.

Introduction

La mise en cache en écriture différée met en file d'attente de manière asynchrone les mises à jour du plug-in Loader. Vous pouvez améliorer les performances en déconnectant les mises à jour, les insertions et les suppressions au sein d'une mappe, le temps système pour la mise à jour de la base de données dorsale. La mise à jour asynchrone est effectuée après un retard (de cinq minutes, par exemple) ou après un certain nombre d'entrées (1 000 entrées).

Lorsque vous configurez le paramètre d'écriture différée dans une mappe de sauvegarde, une unité d'exécution d'écriture différée est créée, qui encapsule le chargeur configuré. Lorsqu'une transaction d'eXtreme Scale insère, met à jour ou supprime une entrée d'une mappe eXtreme Scale, un objet LogElement est créé pour chacun de ces enregistrements. Ces éléments sont envoyés au chargeur en écriture différée et mis en file d'attente dans une mappe ObjectMap spéciale, appelée mappe de files d'attente. Chaque mappe de sauvegarde pour laquelle est

activée l'écriture différée possède ses propres mappes de files d'attente. Une unité d'exécution d'écriture différée supprime régulièrement des données des mappes de files d'attente et les envoie au chargeur dorsal réel.

Le chargeur en écriture différée n'envoie au chargeur réel que les objets LogElement de types insertion, mise à jour et suppression. Tous les autres types d'objets LogElement (EVICT, par exemple) sont ignorés.

La prise en charge de l'écriture différée *est* une extension du plug-in Loader, qui vous permet d'intégrer eXtreme Scale à la base de données. A ce sujet, vous pouvez consulter avec profit les explications «Configuration des chargeurs JPA», à la page 233 sur la configuration d 'un chargeur JPA.

Avantages

L'activation de l'écriture différée présente les avantages suivants :

- Isolement des défaillances du dorsal : la mise en cache en écriture différée fournit une couche d'isolement des défaillances du dorsal. En cas de défaillance de la base de données dorsale , les mises à jour sont mises en file d'attente dans la mappe de files d'attente. Les applications peuvent poursuivre la répartition de transactions dans eXtreme Scale. Une fois le dorsal restauré, les données de la mappe de files d'attente lui sont envoyées.
- Charge dorsale réduite : le chargeur en écriture différée fusionne les mises à jour en fonction de la clé, ce qui fait qu'il n'existe par clé qu'une seule mise à jour fusionnée dans la mappe de files d'attente. Cette fusion réduit le nombre de mises à jour du système dorsal.
- Performances de transaction améliorées : les temps des transactions individuelles d'eXtreme Scale sont réduits, car la transaction n'a pas besoin d'attendre la synchronisation des données avec le système dorsal.

Descripteur ObjectGrid au format XML

Lors de la configuration de eXtreme Scale à l'aide d'un fichier XML de descripteur d'ObjectGrid eXtreme Scale, le chargeur en écriture différée est activé en définissant l'attribut write-behind dans la balise backingMap. Voici un exemple :

```
<objectGrid name="library" >
     <backingMap name="book" writeBehind="T300;C900" pluginCollectionRef="bookPlugins"/>
```

Dans l'exemple ci-dessus, la prise en charge de l'écriture différée par la mappe de sauvegarde "book" est activée avec le paramètre "T300;C900".

L'attribut write-behind spécifie le temps de mise à jour maximal et/ou le nombre de mises à jour de clés maximal. Le format du paramètre write-behind est le suivant :

```
write-behind attribute  ::= <defaults> | <update time> | <update key count> | <update time> ";" <update key count>
update time  ::= "T" <positive integer>
update key count  ::= "C" <positive integer>
defaults  ::= "" {table}
```

Les mises à jour du chargeur ont lieu lorsque l'un des événements suivants se produit :

1. Le temps de mise à jour maximal, en secondes, s'est écoulé depuis la dernière mise à jour.
2. Le nombre de clés mises à jour dans la mappe de files d'attente a atteint le nombre de clés à mettre à jour.

Ces paramètres sont uniquement des indications. Le temps et le nombre réels de mises à jour seront compris dans une plage de paramètres approchants. Toutefois, il n'y a aucune garantie que le délai et le nombre réels des mises à jour soient identiques à ceux définis dans les paramètres. De plus, la première mise à jour suivante est susceptible d'avoir lieu dans un délai supérieur au double de celui de la mise à jour. Cela est dû au fait qu'eXtreme Scale répartit au hasard l'heure de début des mises à jour de manière à ce que toutes les partitions n'accèdent pas à la base de données en même temps.

Dans l'exemple précédent T300;C900, le chargeur écrit les données dans le système dorsal lorsque 300 secondes se sont écoulées depuis la dernière mise à jour ou lorsque 900 clés sont en attente de mise à jour.

Par défaut, le temps de mise à jour est de 300 secondes et le nombre de clés à mettre à jour est de 1 000.

Le tableau ci-dessous répertorie quelques exemples de paramètres d'écriture différée.

Remarque : Si vous configurez le chargeur en écriture différée en tant que chaîne vide : `writeBehind=""`, il est activé avec les valeurs par défaut. Par conséquent, ne spécifiez pas l'attribut d'écriture différée si vous ne souhaitez pas que l'écriture différée soit activée.

Tableau 8. Quelques options d'écriture différée

Valeur d'attribut	Temps
T100	Le temps de mise à jour est de 100 secondes et le nombre de clés à mettre à jour est de 1 000 (valeur par défaut)
C2000	Le temps de mise à jour est de 300 secondes (valeur par défaut) et le nombre de clés à mettre à jour est de 2 000.
T300;C900	Le temps de mise à jour est de 300 secondes et le nombre de clés à mettre à jour est de 900.
""	Le temps de mise à jour est de 300 secondes (valeur par défaut) et le nombre de clés à mettre à jour est de 1 000 (valeur par défaut)

Activation par programmation de l'écriture différée

Lorsque vous créez une mappe de sauvegarde à l'aide d'un programme pour une version d'eXtreme Scale locale en mémoire, vous pouvez utiliser la méthode suivante sur l'interface de BackingMap pour activer ou désactiver l'écriture différée.

```
public void setWriteBehind(String writeBehindParam);
```

Pour en savoir davantage sur l'utilisation de la méthode setWriteBehind, voir dans le *Guide de programmation* les explications concernant l'interface BackingMap..

Considérations liées à la conception d'applications

L'activation de l'écriture différée est une opération simple, mais la création d'une application devant utiliser l'écriture différée requiert une attention particulière. Sans écriture différée, la transaction d'eXtreme Scale inclut la transaction expéditrice. La transaction d'eXtreme Scale démarre avant la transaction expéditrice et se termine après elle.

Lorsque l'écriture différée est activée, la transaction d'eXtreme Scale se termine avant le démarrage de la transaction expéditrice. La transaction d'eXtreme Scale et la transaction expéditrice sont départagées.

Contraintes d'intégrité référentielle

Chaque mappe de sauvegarde configurée avec écriture différée dispose de sa propre unité d'exécution d'écriture différée pour envoyer les données vers le système dorsal. Par conséquent, les données actualisées dans différentes mappes d'une transaction d'eXtreme Scale sont mises à jour sur le dorsal dans les différentes transactions dorsales. Par exemple, la transaction T1 met à jour la clé key1 dans la mappe Map1 et la clé key2 dans la mappe Map2. La clé key1 mise à jour dans la mappe Map1 est actualisée vers le système dorsal dans une transaction expéditrice et la clé key2 actualisée dans la mappe Map2 l'est dans une autre transaction expéditrice ; cette mise à jour vers le dorsal est effectuée dans deux unités d'exécution différentes, toutes deux en écriture différée. Si les données stockées dans Map1 et Map2 ont des liens, tels que des contraintes de clé externe dans le système dorsal, les mises à jour sont susceptibles d'échouer.

Lors de la conception de contraintes d'intégrité référentielle dans votre base de données dorsale, vérifiez que les mises à jour désordonnées sont autorisées.

Echec des mises à jour

La transaction d'eXtreme Scale se terminant avant le démarrage de la transaction expéditrice, il est possible de recevoir un message de réussite de la transaction erroné. Par exemple, si vous tentez d'insérer une entrée dans une transaction d'eXtreme Scale qui n'existe pas dans la mappe de sauvegarde, mais existe bien dans le système dorsal (entraînant ainsi une clé en double), la transaction d'eXtreme Scale réussit. Cependant, la transaction dans laquelle l'unité d'exécution d'écriture différée insère cet objet dans le système dorsal échoue avec une exception de clé en double.

Pour connaître la solution à ces échecs, voir «Traitement des échecs de mises à jour en écriture différée», à la page 130.

Verrouillage d'une mappe de files d'attente

Le comportement de transaction constitue une autre différence fondamentale dans le comportement des transactions. eXtreme Scale prend en charge trois stratégies de verrouillage différentes : PESSIMISTIC, OPTIMISTIC et NONE. La mappe de files d'attente d'écriture différée utilise la stratégie de verrouillage pessimiste, quelle que soit la stratégie configurée pour sa mappe de sauvegarde. Deux types d'opérations acquièrent un verrou sur la mappe de files d'attente :

- Lorsqu'une transaction d'eXtreme Scale est validée, ou lorsqu'un vidage (de mappe ou de session) a lieu, la transaction lit la clé dans la mappe de files d'attente et y place un verrou S.
- Lorsqu'une transaction d'eXtreme Scale est validée, elle tente de mettre à niveau le verrou S vers le verrou X sur la clé.

Ce comportement de mappe de files d'attente supplémentaire vous permet de voir quelques différences dans le comportement de verrouillage.

- Si la mappe de l'utilisateur est configurée comme stratégie de verrouillage pessimiste, la différence dans le comportement de verrouillage n'est pas grande. Chaque fois qu'un vidage ou qu'une validation est appelée, un verrou S est placé sur la même clé dans la mappe de files d'attente. Au moment de la validation, un verrou X est acquis pour la clé dans la mappe de l'utilisateur, mais également pour la clé dans la mappe de files d'attente.

- Si la mappe de l'utilisateur est configurée comme stratégie de verrouillage optimiste ou inexistante, la transaction utilisateur suit le modèle de la stratégie pessimiste. Chaque fois qu'un vidage ou qu'une validation est appelée, un verrou S est acquis sur la même clé dans la mappe de files d'attente. Au moment de la validation, un verrou X est acquis pour la clé dans la mappe de files d'attente utilisant la même transaction.

Nouvelles tentatives de transaction du chargeur

WebSphere eXtreme Scale ne prend pas en charge les transactions à 2 phases ou XA. L'unité d'exécution d'écriture différée supprime les enregistrements de la mappe de files d'attente et met à jour les enregistrements vers le système dorsal. Si le serveur tombe en panne au beau milieu de la transaction, des mises à jour dorsales risquent d'être perdues.

Le chargeur en écriture différée retente automatiquement d'écrire les transactions ayant échouées et envoie un objet LogSequence en attente de validation au système dorsal afin d'éviter la perte de données. Cette action requiert que le chargeur soit idempotent, ce qui signifie que si la méthode Loader.batchUpdate(TxId, LogSequence) est appelée deux fois avec la même valeur, le chargeur doit donner le même résultat que si elle était appliquée une seule fois. Les implémentations du chargeur doivent appliquer l'interface RetryableLoader pour activer cette fonction. Pour plus d'informations, voir dans la documentation d'API.

Echecs du chargeur

Le plug-in du chargeur (Loader) risque d'échouer lorsqu'il ne parvient pas à communiquer avec le dorsal de base de données. Cela peut se produire si le serveur de base de données ou la connexion réseau est arrêté(e). Le chargeur en écriture différée met en file d'attente les mises à jour et tente d'envoyer régulièrement les données modifiées au chargeur. Le chargeur doit avertir WebSphere eXtreme Scale, en phase d'exécution, de l'existence d'un problème de connectivité de la base de données en émettant une exception LoaderNotAvailableException.

Par conséquent, l'implémentation du chargeur doit pouvoir distinguer entre une défaillance de base de données et une défaillance de chargeur physique. Une défaillance de base de données doit être émise ou réémise en tant qu'exception LoaderException ou OptimisticCollisionException, alors qu'une défaillance de chargeur physique doit être émise ou réémise en tant qu'exception LoaderNotAvailableException. WebSphere eXtreme Scale gère ces deux exceptions différemment :

- Si une exception LoaderException est interceptée par le chargeur en écriture différée, celui-ci considère que l'échec est dû à une défaillance de données, telle qu'une erreur de clé en double. Le chargeur dégroupe la mise à jour et tente de ne mettre à jour qu'un enregistrement à la fois, afin d'isoler la défaillance de données. Si une exception LoaderException est à nouveau interceptée lors de la mise à jour d'un enregistrement, un enregistrement de la mise à jour ayant échoué est créé et consigné dans la mappe de mises à jour ayant échoué.
- Si une exception LoaderNotAvailableException est interceptée par le chargeur en écriture différée, celui-ci considère que l'échec est dû à l'impossibilité de se connecter à la base de données, par exemple, lorsque la base de données dorsale est inactive, lorsque la connexion à une base de données est indisponible ou lorsque le réseau est inactif. Le chargeur attend 15 secondes, puis tente à nouveau la mise à jour par lots de la base de données.

L'erreur courante est d'émettre une exception LoaderException à la place d'une exception LoaderNotAvailableException. Dans ce cas, tous les enregistrements mis en file d'attente dans le chargeur en écriture différée deviennent des enregistrements de mise à jour ayant échoué, ce qui rend l'isolation des défaillances dans le système dorsal inutile. Cette erreur est généralement rencontrée lorsque vous écrivez un chargeur générique pour communiquer avec les bases de données.

Le chargeur JPALoader fourni par eXtreme Scale en est un exemple. Le chargeur JPALoader utilise l'API JPA pour interagir avec des bases de données dorsales. Lorsque le réseau échoue, le chargeur JPALoader reçoit une exception javax.persistence.PersitenceException, mais il ignore le motif de l'erreur, sauf si l'état SQL et le code d'erreur SQL de l'exception SQLException associée sont vérifiés. La conception du chargeur JPALoader pour fonctionner avec tout type de base de données complique d'autant plus le problème, car les états et les codes d'erreur SQL sont différents pour un problème d'arrêt de réseau. Pour résoudre ce problème, WebSphere eXtreme Scale fournit un API ExceptionMapper permettant aux utilisateurs d'intégrer une implémentation afin de mapper une exception vers une exception plus facilement utilisable. Par exemple, les utilisateurs peuvent mapper une exception générique javax.persistence.PersitenceException vers une exception LoaderNotAvailableException si l'état ou le code d'erreur SQL indique l'arrêt du réseau.

Remarques sur les performances

En supprimant de la transaction la mise à jour du chargeur, la mise en cache en écriture différée augmente le temps de réponse. Elle accroît également le temps de traitement des bases de données, car les mises à jour des bases de données son combinées. Il est important de bien réaliser le temps système qu'ajoute l'unité d'exécution d'écriture différée, laquelle extrait les données de la mappe de files d'attente et les envoie vers le chargeur.

Le temps de mise à jour maximal et le nombre de mises à jour maximal doivent être ajustés en fonction de l'environnement et des types d'utilisation prévus. Si la valeur du temps ou du nombre de mises à jour maximal est trop petite, le temps système de l'unité d'exécution d'écriture différée peut dépasser les avantages tirés. Définir une valeur élevée pour ces deux paramètres peut également augmenter l'utilisation de la mémoire pour la mise en file d'attente des données et retarder le moment de péremption des enregistrements de bases de données.

Pour des performances optimales, réglez les paramètres d'écriture différée en fonction des facteurs suivants :

- ratio des transactions de lecture et d'écriture
- fréquence de mise à jour d'enregistrements identiques
- temps d'attente pour la mise à jour de la base de données

Traitement des échecs de mises à jour en écriture différée

La transaction d'WebSphere eXtreme Scale se terminant avant le démarrage de la transaction expéditrice, il est possible de recevoir un message de réussite de la transaction erroné.

Par exemple, si vous tentez d'insérer une entrée dans une transaction d'eXtreme Scale qui n'existe pas dans la mappe de sauvegarde, mais existe bien en arrière plan (entraînant ainsi une clé en double), la transaction d'eXtreme Scale réussit.

Cependant, la transaction dans laquelle l'unité d'exécution en écriture différée insère cet objet dans le dorsal échoue avec une exception de clé en double.

Traitement des échecs de mises à jour en écriture différée : côté client

Ce type de mise à jour ou tout autre mise à jour dorsale ayant échoué est une mise à jour en écriture différée échouée. Les échecs de mises à jour en écriture différée sont stockés dans une mappe d'échecs de mises à jour en écriture différée. Cette mappe sert de file d'attente d'événements pour les mises à jour échouées. La clé de la mise à jour est un objet Integer unique et la valeur correspond à une instance de FailedUpdateElement. La mappe d'échecs de mises à jour en écriture différée est configurée avec un expulseur qui expulse les enregistrements une heure après leur insertion. Les enregistrements des échecs de mises à jour sont donc perdus s'ils ne sont pas récupérés dans l'heure.

L'API ObjectMap peut être utilisée pour récupérer les entrées de mappe des échecs de mises à jour en écriture différée. La mappe des échecs de mises à jour en écriture différée s'intitule : IBM_WB_FAILED_UPDATES_<nom mappe>. Pour connaître le nom de préfixe de chacune des mappes système en écriture différée, reportez-vous à la documentation de l'API WriteBehindLoaderConstants. Voici un exemple.

échec de processus - exemple de code

```
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);
    // Faites quelque chose d'intéressant avec la clé, la valeur ou l'exception.

}
session.commit();
```

Un appel getNextKey est compatible avec une partition spécifique pour chaque transaction eXtreme Scale. Dans un environnement réparti, pour obtenir les clés de toutes les partitions, vous devez démarrer plusieurs transactions, tel qu'indiqué dans l'exemple suivant :

obtention des clés de toutes les partitions - exemple de code

```
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);
        // Faites quelque chose d'intéressant avec la clé, la valeur ou l'exception.

    }
    Session.commit();
}
```

Remarque : La mappe d'échecs de mises à jour permet de surveiller l'état de l'application. Si un système produit un grand nombre d'enregistrements dans la mappe d'échecs de mises à jour, c'est l'indice que l'application ou l'architecture ont besoin d'être réévaluées ou qu'elles doivent utiliser l'écriture différée. A partir de la version 6.1.0.5, vous pouvez utiliser le script xsadmin pour afficher la taille des entrées de mappe d'échecs de mises à jour.

Traitement des échecs de mises à jour en écriture différée : programme d'écoute des fragments

La détection et la consignation des échecs de transactions en écriture différée sont importantes. Toutes les applications en écriture différée doivent implémenter un observateur pour traiter les échecs des mises à jour en écriture différée. Cette méthode permet d'éviter d'être à court de mémoire car les enregistrements d'une mappe d'échecs de mises à jour, censés être traités par l'application, ne sont pas expulsés.

Le code suivant montre comment connecter cet observateur ou « videur » qui doit être ajouté au fichier XML du descripteur d'ObjectGrid tel qu'indiqué dans le fragment de code ci-dessous.

```
<objectGrid name="Grid">
 <bean id="ObjectGridEventListener" className="utils.WriteBehindDumper"/>
```

Vous pouvez voir le bean ObjectGridEventListener qui a été ajouté et qui correspond à l'observateur d'écriture différé dont nous avons parlé plus haut. L'observateur interagit avec les mappes pour tous les fragments primaires d'une JVM à la recherche de ceux pour lesquels l'écriture différée a été activée. S'il trouve un fragment, il tente de consigner jusqu'à 100 échecs de mises à jour. Il observe en continu un fragment primaire jusqu'à ce que ce dernier soit déplacé vers une autre JVM. Toutes les applications en écriture différée *doivent* utiliser un observateur similaire à celui-ci. Si ce n'est pas le cas, la mémoire de la machines virtuelles Java devient insuffisante car les enregistrements de cette mappe d'erreurs ne sont jamais expulsés.

Pour plus d'informations, voir Exemple de code de classe dumper en écriture différée .

Exemple de code d'écriture différée d'une classe dumper

Cet exemple de code source montre comment écrire un programme de surveillance (dumper) pour gérer les mises à jour d'écriture différée ayant échoué.

```
//
//Ce programme exemple n'est soumis à aucune redevance ; il est fourni EN L'ETAT et
//peut être librement utilisé, exécuté, copié et modifié par le client (a) dans
//le cadre de son instruction personnelle et de sujets d'étude, (b) pour développer
//des applications conçues pour s'exécuter avec un produit IBM WebSphere ou pour
//être redistribué dans les propres produits du client, dans le cadre de ces
//applications. "
//
//5724-J34 (C) COPYRIGHT International Business Machines Corp. 2009
//  Tous droits réservés *  Eléments sous licence - Propriété d'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;

/**
 * L'écriture différée s'attend à ce que les transactions vers le chargeur aboutissent. Si une
 * transaction échoue pour une clé, elle insère une entrée dans une mappe appelée PREFIXE + nomMappe.
 * L'application doit rechercher dans cette mappe les entrées de vidage des incidents de
 * transaction d'écriture différée. L'application est chargée d'analyser, puis de supprimer
 * ces entrées. Ces dernières peuvent être assez importantes car elles incluent la clé, les
 * images avant et après de la valeur et l'exception elle-même. Les exceptions peuvent facilement
 * atteindre 20k.
 *
 * La classe est enregistrée avec la grille et une instance est créée pour
 * chaque fragment primaire d'une machine virtuelle Java. Une unique unité
 * d'exécution est créée et cette dernière recherche le fragment dans chaque
 * mappe d'erreurs à écriture différée, imprime le problème, puis supprime
 * l'entrée.
 *
 * Cela signifie qu'il existera une unité d'exécution par fragment. Si le
 * fragment est transféré sur une autre machine virtuelle Java, la méthode
 * deactivate arrête l'unité d'exécution.
 * @author bnewport
 *
 */
public class WriteBehindDumper implements ObjectGridEventListener, ObjectGridEventGroup.ShardEvents,
Callable<Boolean>
{
 static Logger logger = Logger.getLogger(WriteBehindDumper.class.getName());

 ObjectGrid grid;

 /**
  * Pool d'unités d'exécution chargé de gérer les vérificateurs de table. Si l'application possède
  * son propre pool, modifiez la valeur pour réutiliser le pool existant
  */
 static ScheduledExecutorService pool = new ScheduledThreadPoolExecutor(2); // deux unités d'exécution

pour vider les enregistrements

 // futur de ce fragment
 ScheduledFuture<Boolean> future;

 // true si ce fragment est actif
 volatile boolean isShardActive;

 /**
  * Durée normale entre les recherches dans les mappes d'erreurs d'écriture différée
  */
 final long BLOCKTIME_SECS = 20L;

 /**
  * Une session allouée pour ce fragment. Il est inutile de les allouer encore et encore
  */
 Session session;
 /**
  * Si un fragment primaire est activé, planifiez les vérifications de sorte à vérifier périodiquement
  * les mappes d'erreurs d'écriture différée et imprimer les éventuels problèmes
  */
 public void shardActivated(ObjectGrid grid)
 {
  try
  {
   this.grid = grid;
   session = grid.getSession();

   isShardActive = true;
   future = pool.schedule(this, BLOCKTIME_SECS, TimeUnit.SECONDS); // vérification initiale

toutes les BLOCKTIME_SECS secondes
  }
  catch(ObjectGridException e)
  {
   throw new ObjectGridRuntimeException("Exception activating write dumper", e);
  }
 }

 /**
  * Marquez le fragment comme inactif, puis annulez le vérificateur
  */
 public void shardDeactivate(ObjectGrid arg0)
 {
  isShardActive = false;
  // s'il est annulé, l'annulation renvoie la valeur true
  if(future.cancel(false) == false)
  {
   // sinon, la tâche est bloquée jusqu'à ce que le vérificateur ait terminé son exécution
```

```
  while(future.isDone() == false) // attendez que la tâche se termine d'une manière ou d'une autre
  {
   try
   {
    Thread.sleep(1000L); // vérifiez à chaque seconde
   }
   catch (InterruptedException e)
   {
   }
  }
 }
}

/**
 * Test simple permettant de vérifier si l'écriture différée est activée
 * pour la mappe ; si c'est le cas, renvoie le nom de la mappe d'erreurs
 * associée.
 * @param mapName Mappe à tester
 * @return Nom de la mappe d'erreurs à écriture différée si elle existe ;
 * sinon null
 */
static public String getWriteBehindNameIfPossible(ObjectGrid grid, String mapName)
{
 BackingMap map = grid.getMap(mapName);
 if(map != null && map.getWriteBehind() != null)
 {
  return WriteBehindLoaderConstants.WRITE_BEHIND_FAILED_UPDATES_MAP_PREFIX + mapName;
 }
 else
  return null;
}

/**
 * Exécuté pour chaque fragment. Vérifie si l'écriture différée est activée pour chaque mappe et
 * si c'est le cas, imprime les éventuelles erreurs de transaction d'écriture différée
 * avant de supprimer l'enregistrement.
 */
public Boolean call()
{
 logger.fine("Called for " + grid.toString());
 try
 {
  // tant que le fragment primaire est présent sur cette machine virtuelle Java,
  // seules les mappes définies par l'utilisateur sont renvoyées ici ; aucune mappe système,
  // telle que les mappes d'écriture différée, ne figure dans cette liste.
  Iterator<String> iter = grid.getListOfMapNames().iterator();
  boolean foundErrors = false;
  // effectuez une itération sur toutes les mappes actuelles
  while(iter.hasNext() && isShardActive)
  {
   String origName = iter.next();

   // s'il s'agit d'une mappe d'erreurs à écriture différée
   String name = getWriteBehindNameIfPossible(grid, origName);
   if (name != null)
   {
    // essayez de supprimer des blocs de N erreurs à la fois
    ObjectMap errorMap = null;
    try
    {
     errorMap = session.getMap(name);
    }
    catch(UndefinedMapException e)
    {
     // au démarrage les mappes d'erreurs risquent de ne pas encore exister ; patience...
     continue;
    }
    // essayez de vider jusqu'à N enregistrements à la fois
    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);
      //
      // Votre application doit consigner le problème ici
      logger.info("WriteBehindDumper ( " + origName + ") for key (" + elem.getKey() + ") Exception: " +
       elem.getThrowable().toString());
      //
      //
      errorMap.remove(seqKey);
     }
     else
      break;
    }
    session.commit();
   }
  } // passez à la mappe suivante
  // bouclez plus rapidement en cas d'erreurs
```

```
   if(isShardActive)
   {
    // replanifiez après une seconde en cas d'enregistrements incorrects ;
    // sinon, attendez 20 secondes.
    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();

   //ne laissez pas de transaction sur la session.
   if(session.isTransactionActive())
   {
    try { session.rollback(); } catch(Exception e2) {}
   }
  }
  return true;
 }

 public void destroy() {
  // TODO Module de remplacement de méthode généré automatiquement

 }

 public void initialize(Session arg0) {
  // TODO Module de remplacement de méthode généré automatiquement

 }

 public void transactionBegin(String arg0, boolean arg1) {
  // TODO Module de remplacement de méthode généré automatiquement

 }

 public void transactionEnd(String arg0, boolean arg1, boolean arg2,
   Collection arg3) {
  // TODO Module de remplacement de méthode généré automatiquement

 }
}
```

Configuration de la réplication entre homologues avec JMS

Le mécanisme de réplication entre homologues basé sur JMS (Java Message Service) est utilisé dans les environnements WebSphere eXtreme Scale réparti et local. JMS est un processus de réplication de coeur à coeur qui permet aux mises à jour de données de circuler parmi les ObjectGrid locaux et les ObjectGrid répartis. Par exemple, avec ce mécanisme, vous pouvez transférer des mises à jour de données d'une grille eXtreme Scale répartie vers une grille eXtreme Scale locale ou d'une grille vers une autre grille d'un autre domaine système.

Avant de commencer

Le mécanisme JMS de réplication entre homologues repose sur l'ObjectGridEventListener JMS pré-intégré, com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener. Pour des informations plus détaillées sur l'activation du mécanisme de réplication entre homologues, voir «Programme d'écoute d'événement JMS», à la page 139.

Pour plus d'informations, voir «Activation du mécanisme d'invalidation du client», à la page 212.

Vous trouverez ci-après un exemple de configuration XML permettant d'activer un mécanisme de réplication entre homologues sur une configuration eXtreme Scale :

Configuration de réplication entre homologues - Exemple 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>
```

Répartir les modifications entre des machines virtuelles Java homologues

Les objets LogSequence et LogElement répartissent les modifications entre des machines virtuelles Java homologues et, à l'aide d'un plug-in ObjectGridEventListener, ils communiquent les modifications intervenues dans une transaction eXtreme Scale.

Pour plus d'informations sur la manière dont Java Message Service (JMS) peut être utilisé pour répartir des modifications transactionnelles, voir dans *Présentation du produit* les explications concernant l'utilisation de JMS pour répartir des modifications transactionnelles.

Il est impératif au préalable que l'instance ObjectGrid soit mise en cache par ObjectGridManager. Voir les méthodes createObjectGrid pour plus d'informations à ce sujet. La valeur booléenne de cacheInstance doit être définie comme true.

Il n'est pas nécessaire que vous implémentiez vous-même ce mécanisme. Il existe en effet un mécanisme pré-intégré de réplication entre homologues qui se chargera de cette fonction. Voir dans le *Guide d'administration* les explications concernant la configuration de la réplication entre homologues avec JMS.

Les objets fournissent aux applications le moyen de publier facilement les modifications qui sont intervenues dans un ObjectGrid, à savoir un transport de messages vers des ObjectGrids homologues situés sur des machines virtuelles Java pour appliquer ces modifications sur ces machines virtuelles Java. La classe LogSequenceTransformer est indispensable pour cette prise en charge. Nous allons expliquer ici comment écrire, pour la propagation des messages, un programme d'écoute utilisant un système de messagerie Java Message Service (JMS). En fait un plug-in IBM permet à eXtreme Scale de prendre en charge la transmission de LogSequences qui résultent de la validation d'une transaction eXtreme Scale entre des membres d'un cluster WebSphere Application Server. Cette fonction n'est pas activée par défaut, mais il est possible de la configurer pour la rendre opérationnelle. Mais, lorsque l'utilisateur ou le producteur ne sont pas WebSphere Application Server, le recours à un système externe de messagerie JMS peut s'avérer nécessaire.

Implémenter le mécanisme

La classe LogSequenceTransformer et les API ObjectGridEventListener, LogSequence et LogElement permettent l'utilisation de n'importe quel mécanisme de publication/abonnement pour la répartition des modifications et le filtrage des mappes à répartir. Les fragments de code utilisés ici montrent comment exploiter ces API avec JMS pour construire un ObjectGrid d'égal à égal, partagé par des applications hébergées sur plusieurs sortes de plateformes partageant un transport commun de messages.

Initialisation du plug-in

L'ObjectGrid appelle la méthode initialize du plug-in, qui fait partie du contrat de l'interface ObjectGridEventListener, lorsque l'ObjectGrid démarre. La méthode initialize doit obtenir ses ressources JMS (connexions, sessions et diffuseurs de publications) et elle doit démarrer l'unité d'exécution qu'est le programme d'écoute JMS.

Les exemples suivants illustrent la méthode initialize :

exemple de méthode 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();

            // need to start the connection to receive messages.
            connection.start();

            // the jms session is not transactional (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);
            // start the listener thread.
            listenerRunning = true;
            listenerThread = new Thread(this);
            listenerThread.start();
        } catch (Throwable e) {
            throw new ObjectGridRuntimeException("Cannot initialize", e);
        }
    }
```

L'unité d'exécution lancée par le code est une unité Java 2 Platform, Standard Edition (Java SE). Pour une exécution sur WebSphere Application Server version 6.x ou sur WebSphere Application Server version 5.x Enterprise, vous devrez utiliser l'API de bean asynchrone pour lancer cette unité d'exécution de démon. Vous pouvez également utiliser les API communes. Voici un exemple de fragment de code montrant la même action effectuée à l'aide d'un gestionnaire de travaux :

```
// start the listener thread.
listenerRunning = true;
workManager.startWork(this, true);
```

Par ailleurs, le plug-in doit implémenter l'interface Work et non l'interface Runnable. Vous devez également ajouter une méthode release pour définir comme false la variable listenerRunning. Le plug-in doit être fourni avec une instance WorkManager dans son constructeur ou par injection si l'on utilise un conteneur IoC (Inversion of Control).

Transmission des modifications

Voici un exemple de méthode transactionEnd pour la publication des modifications locales apportées à un ObjectGrid. Cet exemple utilise JMS, bien qu'il soit possible d'utiliser n'importe quel transport de messages capable de publication/abonnement fiable.

Exemple de méthode transactionEnd

```
// This method is synchronized to make sure the
    // messages are published in the order the transaction
    // were committed. If we started publishing the messages
    // in parallel then the receivers could corrupt the Map
    // as deletes may arrive before inserts etc.
    public synchronized void transactionEnd(String txid, boolean isWriteThroughEnabled,
    boolean committed,
            Collection changes) {
        try {
            // must be write through and commited.
            if (isWriteThroughEnabled && committed) {
                // write the sequences to a byte []
                ByteArrayOutputStream bos = new ByteArrayOutputStream();
                ObjectOutputStream oos = new ObjectOutputStream(bos);
                if (publishMaps.isEmpty()) {
                    // serialize the whole collection
                    LogSequenceTransformer.serialize(changes, oos, this, mode);
                } else {
                    // filter LogSequences based on publishMaps contents
                    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);
                }
                // make an object message for the changes
                oos.flush();
                ObjectMessage om = jmsSession.createObjectMessage(bos.toByteArray());
                // set properties
                om.setStringProperty(PROP_TX, txid);
                om.setStringProperty(PROP_GRIDNAME, myGrid.getName());
                // transmit it.
                publisher.publish(om);
            }
        } catch (Throwable e) {
            throw new ObjectGridRuntimeException("Cannot push changes", e);
        }
    }
```

Cette méthode utilise plusieurs variables d'instance :

- La variable `jmsSession` : session JMS servant à publier les messages. Elle est créée lors de l'initialisation du plug-in.
- La variable `mode` : le mode de répartition.
- La variable `publishMaps` : ensemble contenant le nom de chacune des mappes dont les modifications sont à publier. La variable vide signifie que la totalité des mappes sont à publier.
- La variable `publisher` : objet TopicPublisher qui est créé durant l'initialisation du plug-in.

Réception et application des messages d'actualisation

Vient à présent la méthode run. Cette méthode s'exécute en boucle jusqu'à ce que l'application arrête la boucle. Chaque itération de la boucle tente de réceptionner un message JMS et de l'appliquer à l'ObjectGrid.

Exemple de méthode run de message JMS

```
private synchronized boolean isListenerRunning() {
        return listenerRunning;
    }
```

```
public void run () {
    try {
        System.out.println("Listener starting");
        // get a jms session for receiving the messages.
        // Non transactional.
        TopicSession myTopicSession;
        myTopicSession = connection.createTopicSession(false, javax.jms.
   Session.AUTO_ACKNOWLEDGE);

        // get a subscriber for the topic, true indicates don't receive
        // messages transmitted using publishers
        // on this connection. Otherwise, we'd receive our own updates.
        TopicSubscriber subscriber = myTopicSession.createSubscriber(topic,
   null, true);
        System.out.println("Listener started");
        while (isListenerRunning()) {
            ObjectMessage om = (ObjectMessage) subscriber.receive(2000);
            if (om != null) {
                // Use Session that was passed in on the initialize...
                // very important to use no write through here
                mySession.beginNoWriteThrough();
                byte[] raw = (byte[]) om.getObject();
                ByteArrayInputStream bis = new ByteArrayInputStream(raw);
                ObjectInputStream ois = new ObjectInputStream(bis);
                // inflate the LogSequences
                Collection collection = LogSequenceTransformer.inflate(ois,
       myGrid);
                Iterator iter = collection.iterator();
                while (iter.hasNext()) {
                    // process each Maps changes according to the mode when
                    // the LogSequence was serialized
                    LogSequence seq = (LogSequence) iter.next();
                    mySession.processLogSequence(seq);
                }
                mySession.commit();
            } // if there was a message
        } // while loop
        // stop the connection
        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");
}
```

Programme d'écoute d'événement JMS

Le programme JMSObjectGridEventListener est conçu pour prendre en charge l'invalidation du cache local côté client et un mécanisme de réplication entre homologues. Il s'agit d'une implémentation JMS (Java Message Service) de l'interface ObjectGridEventListener.

Le mécanisme d'invalidation client peut être utilisé dans un environnement eXtreme Scale réparti pour garantir la synchronisation des données du cache local avec les serveurs ou les autres clients. Sans cette fonction, le cache local du client pourrait contenir des données obsolètes. Toutefois, même avec ce mécanisme d'invalidation client JMS, vous devez prendre en compte le délai de mise à jour d'un cache local client en raison du retard de la publication des mises à jour.

Le mécanisme de réplication entre homologues peut être utilisé dans les environnements eXtreme Scale répartis et locaux. Il s'agit d'un processus de réplication de coeur à coeur qui permet aux mises à jour de données de circuler parmi les ObjectGrid locales et les ObjectGrid réparties. Par exemple, avec ce mécanisme, vous pouvez transférer des mises à jour de données d'une grille répartie vers une grille locale ou d'une grille vers une autre grille d'un autre domaine système.

Le programme JMSObjectGridEventListener exige de l'utilisateur la configuration des informations JMS et JNDI (Java Naming and Directory Interface) pour obtenir les ressources JMS requises. En outre, les propriétés de réplication doivent être définies correctement. Dans un environnement JEE, les informations JNDI doivent être disponibles dans les conteneurs Web et EJB (Enterprise JavaBean). Dans ce cas, la propriété JNDI est facultative à moins que vous ne souhaitiez obtenir des ressources JMS externes.

Ce programme d'écoute d'événement comporte des propriétés que vous pouvez configurer avec le langage XML ou à l'aide de programmes et pouvant être utilisées pour l'invalidation client, pour la réplication entre homologues ou les deux. La plupart des propriétés sont facultatives afin de personnaliser le comportement permettant d'obtenir les fonctionnalités dont vous avez besoin.

.

Pour plus d'informations, consultez l'API JMSObjectGridEventListener.

Extension du plug-in JMSObjectGridEventListener

Le plug-in JMSObjectGridEventListener permet aux instances ObjectGrid homologues de recevoir des mises à jour lorsque les données de la grille sont modifiées ou expulsées. Il permet également aux clients d'être avertis lors de la mise à jour ou de l'expulsion d'entrées d'une grille eXtreme Scale. Cette rubrique décrit l'extension du plug-in JMSObjectGridEventListener pour permettre aux applications d'obtenir une notification à réception d'un message JMS. Cette fonction est particulièrement utile lors de l'utilisation du paramètre CLIENT_SERVER_MODEL pour l'invalidation client.

Lors d'une exécution avec le rôle récepteur, la méthode JMSObjectGridEventListener.onMessage substituée est automatiquement appelée par l'exécution eXtreme Scale lorsque l'instance JMSObjectGridEventListener reçoit des mises à jour du message JMS de la grille. Ces messages incluent une collection d'objets LogSequence. Les objets LogSequence sont transmis à la méthode onMessage et l'application utilise l'objet LogSequence pour identifier les entrées de cache qui ont été insérées, supprimées, mises à jour ou invalidées.

Pour utiliser le point d'extension onMessage, les applications suivent les étapes ci-dessous.

1. Crée une classe, en étendant la classe JMSObjectGridEventListener et en substituant la méthode onMessage.
2. Configure la classe étendue JMSObjectGridEventListener de la même manière que la classe ObjectGridEventListener pour l'ObjectGrid.

La classe étendue JMSObjectGridEventListener est un enfant de la classe JMSObjectGridEventListener et peut uniquement se substituer à deux méthodes : les méthodes initialize (facultative) et onMessage. Si une classe enfant de la classe

JMSObjectGridEventListener doit utiliser des artefacts ObjectGrid tels que ObjectGrid ou Session dans la méthode onMessage, elle peut obtenir ces artefacts dans la méthode initialize et les mettre en cache en tant que variables d'instance. De même, dans la méthode onMessage, les artefacts ObjectGrid mis en cache doivent être utilisés pour traiter une collection de LogSequences transmise.

Remarque : La méthode initialize remplacée doit appeler la méthode super.initialize pour initialiser la classe parent JMSObjectGridEventListener de manière appropriée.

Voici un exemple de classe étendue JMSObjectGridEventListener.

```
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;

    /**
     * Grille associée à ce programme d'écoute d'événement.
     */
    ObjectGrid grid;

    /**
     * Session associée à ce programme d'écoute d'événement.
     */
    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) {
  // Remarque : si l'utilisation d'un artefact ObjectGrid est requise,
            //  cette classe doit obtenir l'ObjectGrid
  //  de l'instance Session transmise et obtenir l'ObjectMap de l'instance Session
  // pour toutes les opérations de mappe ObjectGrid transactionnelles.

  super.initialize(session);  // doit appeler la méthode initialize super.
  this.session = session; // mettez en cache l'instance de session si son utilisation est
  //   requise pour effectuer une opération de mappe.
  this.grid = session.getObjectGrid(); // obtenez ObjectGrid, si vous devez obtenir
  //   des informations d'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 + "]";
  }
}
```

Configuration

La classe étendue JMSObjectGridEventListener doit être configurée de la même manière pour le mécanisme d'invalidation client que pour le mécanisme de réplication entre homologues. L'exemple suivant illustre l'approche de la configuration 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>
```

Remarque : Le nom de classe du bean ObjectGridEventListener est configuré à l'aide de la classe étendue JMSObjectGridEventListener avec les mêmes propriétés que la classe générique JMSObjectGridEventListener.

Fichier XML du descripteur d'ObjectGrid

Pour configurer WebSphere eXtreme Scale, utilisez un fichier XML de descripteur d'ObjectGrid et l'API ObjectGrid.

Dans les sections ci-après, des exemples de fichier XML sont fournis pour illustrer les diverses configurations. Chaque élément et attribut du fichier XML est défini. Utilisez le schéma du XML du descripteur d'ObjectGrid pour créer le fichier XML du descripteur. Pour un exemple de schéma XML de descripteur d'ObjectGrid, voir la rubrique «Fichier objectGrid.xsd», à la page 160.

Une version modifiée du fichier companyGrid.xml d'origine est utilisée. Le fichier companyGridSingleMap.xml ci-après ressemble au fichier companyGrid.xml. Le fichier companyGridSingleMap.xml contient une mappe et le fichier companyGrid.xml en contient quatre. Les éléments et attributs du fichier sont décrits en détails après l'exemple.

```
<?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>
```

Elément objectGridConfig

L'élément objectGridConfig correspond à l'élément de niveau supérieur du fichier XML. Enregistrez cet élément dans votre document XML eXtreme Scale, comme illustré dans l'exemple précédent. Cet élément configure l'espace de noms du fichier et l'emplacement du schéma. Le schéma est défini dans le fichier objectGrid.xsd.

- nombre d'occurrences : une
- élément enfant : élément objectGrids et élément backingMapPluginCollections

Elément objectGrids

L'élément objectGrids sert de conteneur à tous les éléments objectGrid. Dans le fichier companyGridSingleMap.xml, l'élément objectGrids contient un objectGrid, l'objectGrid CompanyGrid.

- nombre d'occurrences : une ou plusieurs
- élément enfant : élément objectGrid

Elément objectGrid

Utilisez l'élément objectGrid pour définir un ObjectGrid. Chacun des attributs de l'élément objectGrid correspond à une méthode de l'interface ObjectGrid.

- nombre d'occurrences : une à plusieurs
- élément enfant : élément bean, élément backingMap, élément querySchema et élément streamQuerySet

Attributs

name
Indique le nom affecté à l'ObjectGrid. La validation XML échoue si cet attribut est manquant. (Obligatoire)

securityEnabled
Active la sécurité au niveau ObjectGrid, qui active les autorisations d'accès aux données de la mappe, lorsque vous affectez à l'attribut la valeur `true`. La valeur par défaut est `true` (facultatif).

authorizationMechanism
Définit le mécanisme d'autorisation de l'élément. Vous pouvez affecter à cet attribut une ou plusieurs valeurs : `AUTHORIZATION_MECHANISM_JAAS` ou `AUTHORIZATION_MECHANISM_CUSTOM`. La valeur par défaut est `AUTHORIZATION_MECHANISM_JAAS`. Spécifiez `AUTHORIZATION_MECHANISM_CUSTOM` si vous utilisez un plug-in MapAuthorization personnalisé. Vous devez affecter à l'attribut securityEnabled la valeur `true` pour que l'attribut authorizationMechanism soit appliqué (facultatif).

permissionCheckPeriod
Spécifie un nombre entier de secondes qui indique la fréquence de vérification des droits d'accès utilisés pour autoriser l'accès d'un client. La valeur par défaut est `0`. Si vous affectez à l'attribut la valeur `0`, chaque appel de méthode get, put, update, remove ou evict demande au mécanisme d'autorisation, une autorisation JAAS (Java Authentication and Authorization Service) ou une autorisation personnalisée, de vérifier si le sujet actuel possède les droits requis. Une valeur supérieure à 0 indique le nombre de secondes pendant lequel un ensemble de droits d'accès doit être placé en cache avant de retourner au mécanisme d'autorisation pour être régénéré. Vous devez affecter à l'attribut securityEnabled la valeur `true` pour que l'attribut permissionCheckPeriod soit appliqué (facultatif).

txTimeout
Indique le délai d'exécution maximal autorisé pour une transaction. Si une transaction n'est pas terminée une fois ce délai écoulé, elle est marquée pour annulation et une exception TransactionTimeoutException est générée. Si vous spécifiez la valeur `0`, les transactions n'expirent jamais (facultatif).

entityMetadataXMLFile
Indique le chemin relatif du fichier XML du descripteur d'entité. Ce chemin est relatif par rapport à l'emplacement du fichier du descripteur Objectgrid. Utilisez cet attribut pour définir un schéma d'entité à l'aide d'un fichier XML. Les entités doivent être définies avant le démarrage d'eXtreme Scale pour que chacune puisse être associée à une mappe de sauvegarde (facultatif).

```
<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"
/>
```

Dans l'exemple ci-après, le fichier `companyGridObjectGridAttr.xml` illustre une manière de configurer les attributs d'un élément objectGrid. La sécurité est activée, le mécanisme d'autorisation est JAAS et la période de vérification des droits d'accès est de 45 secondes. Le fichier enregistre également les entités en spécifiant un attribut 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>
```

L'exemple de code suivant illustre l'approche par programme permettant d'obtenir la même configuration qu'avec le fichier `companyGridObjectGridAttr.xml` de l'exemple précédent.

```
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"));
```

Elément backingMap

L'élément backingMap est utilisé pour définir une instance BackingMap sur un ObjectGrid. Chacun des attributs de l'élément backingMap correspond à une méthode de l'interface BackingMap. Pour les détails, voir dans le *Guide de programmation* les explications concernant l'interface BackingMap.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : élément timeBasedDBUpdate

Attributs

name
Indique le nom attribué à l'instance backingMap. Si cet attribut est manquant, la validation XML échoue. (Obligatoire)

readOnly
Définit une instance BackingMap comme étant en lecture seule si vous spécifiez la valeur `false` pour cet attribut. Si vous affectez à l'attribut la valeur `true`, l'instance BackingMap est en lecture seule. Les appels à Session.getMap(String) entraînent la création de mappes dynamiques si le nom transmis à la méthode correspond à l'expression régulière spécifiée dans l'attribut name de cette mappe de sauvegarde. La valeur par défaut est `false` (facultatif).

template
Indique si des mappes dynamiques peuvent être utilisées. Spécifiez `true` si la mappe BackingMap est un modèle de mappe. Les modèles de mappe peuvent être utilisés pour créer dynamiquement des mappes après le démarrage d'ObjectGrid. Les appels à Session.getMap(String) entraînent la création de mappes dynamiques si le nom transmis à la méthode correspond à l'expression régulière spécifiée dans l'attribut name de la mappe de sauvegarde. La valeur par défaut est `false` (facultatif).

pluginCollectionRef
Indique une référence à un plug-in backingMapPluginCollection. La valeur de cet attribut doit correspondre à l'attribut ID d'un plug-in

backingMapCollection. La validation échoue s'il n'existe aucun ID correspondant. Définissez l'attribut pour réutiliser les plug-in BackingMap (facultatif).

numberOfBuckets
Indique le nombre de compartiments à utiliser par l'instance BackingMap. L'instance BackingMap utilise une mappe de hachage pour l'implémentation. S'il existe plusieurs entrées dans la mappe de sauvegarde, plus les compartiments sont nombreux, plus les performances sont meilleures car le risque de collision diminue lorsque le nombre de compartiments augmente. Plus les compartiments sont nombreux, plus les accès simultanés le sont également. Spécifiez la valeur `0` pour désactiver le cache local sur un client lors de la communication à distance avec eXtreme Scale. Si vous spécifiez la valeur `0` pour un client, ne spécifiez la valeur que dans le fichier du descripteur XML ObjectGrid de remplacement par les clients (facultatif).

preloadMode
Définit le mode de préchargement si un plug-in de programme de chargement est défini pour cette instance BackingMap. La valeur par défaut est `false`. Si l'attribut possède la valeur `true`, la méthode Loader.preloadMap(Session, BackingMap) est appelée de manière asynchrone. Sinon, l'exécution de la méthode est bloquée lors du chargement des données pour que le cache ne soit pas disponible avant la fin du préchargement. Le préchargement a lieu au cours de l'initialisation (facultatif).

lockStrategy
Indique si le gestionnaire de verrouillage interne est utilisé chaque fois qu'une transaction accède à une entrée de mappe. Spécifiez l'une des trois valeurs suivantes pour cet attribut : `OPTIMISTIC`, `PESSIMISTIC` ou `NONE`. La valeur par défaut est `OPTIMISTIC` (facultatif).

La stratégie de verrouillage optimiste est généralement utilisée lorsqu'une mappe ne possède pas de plug-in de chargeur, qu'elle est la plupart du temps lue, mais rarement éditée ou mise à jour et que le verrouillage n'est pas fourni par le gestionnaire de persistance avec l'utilisation d'eXtreme Scale comme cache secondaire ou par l'application. Un verrou exclusif est obtenu sur une entrée de mappe insérée, mise à jour ou supprimée lors de la phase de validation. Le verrou garantit que les informations de version ne peuvent pas être modifiées par une autre transaction alors que la transaction en cours de validation effectue une vérification de version optimiste.

La stratégie de verrouillage pessimiste est généralement utilisée pour une mappe qui ne possède pas de plug-in de chargeur, quand le verrouillage n'est pas assuré par un gestionnaire de persistance avec l'utilisation d'eXtreme Scale comme cache secondaire, par un plug-in de chargeur ou par l'application. La stratégie de verrouillage pessimiste est utilisée en cas d'échecs trop fréquents de la stratégie de verrouillage optimiste car les transactions de mise à jour entrent souvent en collision sur la même entrée de mappe.

La stratégie sans verrouillage indique que le gestionnaire de verrouillage interne n'est pas requis. Le contrôle des accès simultanés est fourni en dehors d'eXtreme Scale, par le gestionnaire de persistance qui utilise eXtreme Scale comme application ou cache secondaire ou par le plug-in du chargeur qui utilise les verrous de base de données pour contrôler les accès simultanés.

Pour plus d'informations, voir dans le *Guide de programmation* les explications concernant le verrouillage des entrées de mappe.

numberOfLockBuckets
Définit le nombre de compartiments de verrouillage utilisés par le gestionnaire

de verrouillage pour l'instance BackingMap. Affectez à l'attribut lockStrategy la valeur OPTIMISTIC ou PESSIMISTIC afin de créer un gestionnaire de verrouillage pour l'instance BackingMap. Le gestionnaire de verrouillage utilise une mappe de hachage pour rechercher les entrées verrouillées par une ou plusieurs transactions. S'il existe de nombreuses entrées, plus les compartiments de verrouillage sont nombreux, plus les performances sont meilleures car le risque de collision diminue lorsque le nombre de compartiments augmente. Plus les compartiments de verrouillage sont nombreux, plus les accès simultanés le sont également. Affectez à l'attribut lockStrategy la valeur NONE pour indiquer à l'instance BackingMap de ne pas utiliser de gestionnaire de verrouillage (facultatif).

lockTimeout
Définit le délai de verrouillage utilisé par le gestionnaire de verrouillage pour l'instance BackingMap. Affectez à l'attribut lockStrategy la valeur OPTIMISTIC ou PESSIMISTIC afin de créer un gestionnaire de verrouillage pour l'instance BackingMap. Pour empêcher les interblocages, le gestionnaire de verrouillage possède un délai d'expiration par défaut de 15 secondes. Si le délai d'expiration est dépassé, une exception LockTimeoutException est générée. La valeur par défaut de 15 secondes suffit pour la plupart des applications, mais sur un système particulièrement chargé, le délai d'expiration peut être dépassé en l'absence d'interblocage. Utilisez l'attribut lockTimeout pour spécifier une valeur supérieure à la valeur par défaut afin d'empêcher les fausses exceptions de délai d'expiration dépassé. Affectez à l'attribut lockStrategy la valeur NONE pour indiquer à l'instance BackingMap de ne pas utiliser de gestionnaire de verrouillage (facultatif).

CopyMode
Indique si une opération get d'une entrée de l'instance BackingMap renvoie la valeur réelle, une copie de la valeur ou un proxy de la valeur. Affectez à l'attribut CopyMode l'une des cinq valeurs suivantes :

COPY_ON_READ_AND_COMMIT
La valeur par défaut est COPY_ON_READ_AND_COMMIT. Spécifiez la valeur COPY_ON_READ_AND_COMMIT pour qu'une application ne fasse jamais référence à l'objet de valeur qui se trouve dans l'instance BackingMap. A la place, l'application utilise toujours une copie de la valeur qui se trouve dans l'instance BackingMap (facultatif).

COPY_ON_READ
Spécifiez la valeur COPY_ON_READ pour améliorer les performances par rapport à la valeur COPY_ON_READ_AND_COMMIT en éliminant la copie créée lors de la validation d'une transaction. Pour conserver l'intégrité des données de la mappe de sauvegarde, l'application s'engage à supprimer toutes les références à une entrée une fois que la transaction est validée. Si vous définissez cette valeur, une méthode ObjectMap.get renvoie une copie de la valeur au lieu d'une référence à la valeur, ce qui garantit que les modifications apportées par l'application à la valeur n'affectent pas l'élément BackingMap tant que la transaction n'est pas validée.

COPY_ON_WRITE
Spécifiez la valeur COPY_ON_WRITE pour améliorer les performances par rapport à la valeur COPY_ON_READ_AND_COMMIT en éliminant la copie créée lors du premier appel de la méthode ObjectMap.get par une transaction pour une clé donnée. A la place, la méthode ObjectMap.get renvoie un proxy de la valeur au lieu d'une référence directe à l'objet

de valeur. Le proxy garantit qu'aucune copie de la valeur n'est effectuée tant que l'application n'appelle pas de méthode set sur l'interface de la valeur.

NO_COPY
Spécifiez la valeur NO_COPY pour permettre à une application de ne jamais modifier d'objet de valeur obtenu à l'aide d'une méthode ObjectMap.get en échange de meilleures performances. Spécifiez la valeur NO_COPY pour les mappes associées aux entités de l'API EntityManager.

COPY_TO_BYTES
Spécifiez la valeur COPY_TO_BYTES pour améliorer l'encombrement mémoire des objets de type complexe et les performances lorsque la copie d'un objet s'appuie sur la sérialisation. Si un objet ne peut pas être cloné ou qu'aucune interface ObjectTransformer personnalisée avec une méthode copyValue efficace n'est fournie, le mécanisme de copie par défaut doit sérialiser et inflatez l'objet pour effectuer une copie. Avec le paramètre COPY_TO_BYTES, l'inflation n'est effectuée que lors d'une opération de lecture et la sérialisation, lors d'une validation.

Pour plus d'informations sur ces paramètres, voir les informations sur les meilleures pratiques de CopyMode dans le *Guide de programmation*..

valueInterfaceClassName
Indique une classe requise lorsque vous affectez à l'attribut CopyMode la valeur COPY_ON_WRITE. Cet attribut est ignoré pour tous les autres modes. La valeur COPY_ON_WRITE utilise un proxy lors des appels de méthode ObjectMap.get. Le proxy garantit qu'aucune copie de la valeur n'est effectuée tant que l'application n'appelle pas de méthode set sur la classe spécifiée comme attribut valueInterfaceClassName (facultatif).

copyKey
Indique si la copie de la clé est requise lors de la création d'une entrée de mappe. La copie de l'objet de clé permet à l'application d'utiliser le même objet de clé pour chaque opération ObjectMap. Spécifiez la valeur true pour copier l'objet de clé lors de la création d'une entrée de mappe. La valeur par défaut est false (facultatif).

nullValuesSupported
Spécifiez la valeur true pour prendre en charge les valeurs null dans la mappe d'objets. Si les valeurs null sont admises, une opération get qui renvoie la valeur null peut indiquer que la valeur est null ou que la mappe ne contient pas la clé transmise à la méthode. La valeur par défaut est true (facultatif).

ttlEvictorType
Indique comment est calculée l'heure d'expiration d'une entrée BackingMap. Donnez à cet attribut l'une de ces valeurs : CREATION_TIME, LAST_ACCESS_TIME, LAST_UPDATE_TIME ou NONE. La valeur CREATION_TIME indique que l'heure d'expiration de l'entrée correspond à la somme de l'heure de création de l'entrée et de la valeur de l'attribut timeToLive. La valeur LAST_ACCESS_TIME indique que l'heure d'expiration de l'entrée correspond à la somme de l'heure du dernier accès à l'entrée (que celle-ci ait été modifiée ou simplement lue) et de la valeur de l'attribut timeToLive. La valeur LAST_UPDATE_TIME indique que l'heure d'expiration de l'entrée correspond à la somme de l'heure de la dernière modification de l'entrée et de la valeur de l'attribut timeToLive. La valeur NONE, qui est la valeur par défaut, indique que l'entrée ne possède pas d'heure d'expiration et qu'elle est présente dans l'instance BackingMap jusqu'à ce que l'application la supprime ou l'invalide de manière explicite (facultatif).

timeToLive
Indique en secondes combien de temps chaque entrée de mappe est présente. La valeur par défaut `0` signifie que l'entrée de mappe est présente indéfiniment ou jusqu'à ce que l'application la supprime ou l'invalide de manière explicite. Sinon, l'expulseur TTL expulse l'entrée de mappe en fonction de cette valeur (facultatif).

streamRef
Indique que la mappe de sauvegarde est une mappe source de flux. Toute insertion ou mise à jour de la mappe de sauvegarde est convertie en un événement de flux de données vers le moteur de requête de flux. Cet attribut doit faire référence à un nom de flux valide dans un ensemble de requêtes de flux (facultatif).

viewRef
Indique que la mappe de sauvegarde est une mappe de vue. La sortie de la vue du moteur de requête de flux est convertie au format de nuplet eXtreme Scale et placée dans la mappe (facultatif).

writeBehind
Indique que l'écriture différée est prise en charge avec les paramètres write-behind (facultatif). Les paramètres write-behind consistent en une durée maximale de mise à jour et un nombre maximal de mises à jour de clés. Leur format est `"[T(time)][;][C(count)]"`. La base de données est mise à jour si l'un des événements suivants se produit :

- La durée de mise à jour maximale, spécifiée en secondes, est dépassée depuis la dernière mise à jour.
- Le nombre de mises à jour disponibles dans la mappe de files d'attente a atteint le nombre maximal de mises à jour.

Pour plus d'informations, voir «Prise en charge de la mise en cache en écriture différée», à la page 125.

La prise en charge de l'écriture différée est une extension du plug-in Loader, qui vous permet d'intégrer eXtreme Scale à la base de données. A ce sujet, vous pouvez consulter avec profit les explications «Configuration des chargeurs JPA», à la page 233 sur la configuration d'un chargeur JPA.

evictionTriggers
Définit les types de déclencheur d'expulsion supplémentaires à utiliser. Tous les expulseurs de la mappe de sauvegarde utilisent cette liste de déclencheurs supplémentaires. Pour éviter une exception IllegalStateException, cet attribut doit être appelé avant la méthode ObjectGrid.initialize(). En outre, notez que la méthode ObjectGrid.getSession() appelle la méthode ObjectGrid.initialize() de manière implicite si cette dernière doit toujours être appelée par l'application. Les entrées de la liste de déclencheurs sont séparées par des points-virgules. Les déclencheurs d'expulsion actuels incluent MEMORY_USAGE_THRESHOLD (facultatif).

```
<backingMap
(1)   name="objectGridName"
(2)   readOnly="true" | "false"
(3)   template="true" | "false"
(4)   pluginCollectionRef="reference to backingMapPluginCollection"
(5)   numberOfBuckets="number of buckets"
(6)   preloadMode="true" | "false"
(7)   lockStrategy="OPTIMISTIC" | "PESSIMISTIC" | "NONE"
(8)   numberOfLockBuckets="number of lock buckets"
(9)   lockTimeout="lock timeout"
(10)   copyMode="COPY_ON_READ_AND_COMMIT" | "COPY_ON_READ" | "COPY_ON_WRITE"
       | "NO_COPY" | "COPY_TO_BYTES"
(11)   valueInterfaceClassName="value interface class name"
(12)   copyKey="true" | "false"
(13)   nullValuesSupported="true" | "false"
```

```
(14)   ttlEvictorType="CREATION_TIME" | "LAST_ACCESS_TIME" | "LAST_UPDATE_TIME" | NONE"
(15)   timeToLive="time to live"
(16)   streamRef="reference to a stream"
(17)   viewRef="reference to a view"
(18)   writeBehind="write-behind parameters"
(19)   evictionTriggers="MEMORY_USAGE_THRESHOLD"
/>
```

Dans l'exemple ci-après, le fichier companyGridBackingMapAttr.xml permet d'illustrer un exemple de configuration de mappe de sauvegarde.

```
<?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>
```

L'exemple de code suivant illustre l'approche par programme permettant d'obtenir la même configuration qu'avec le fichier companyGridBackingMapAttr.xml de l'exemple précédent :

```
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);

// Si vous spécifiez le mode de copie COPY_ON_WRITE, une classe valueInterface est requise
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); // spécifie une durée de vie de 50 minutes
```

Elément bean

Utilisez l'élément bean pour définir des plug-in. Vous pouvez connecter des plug-in à des éléments objectGrid et BackingMap.

- nombre d'occurrences dans l'élément objectGrid : zéro à plusieurs
- nombre d'occurrences dans l'élément backingMapPluginCollection : zéro à plusieurs
- élément enfant : élément property

Attributs

id Indique le type de plug-in à créer. (Obligatoire)

Les plug-in valides pour un bean qui est un élément enfant de l'élément objectGrid sont inclus dans la liste suivante :

- Plug-in TransactionCallback
- Plug-in ObjectGridEventListener

- Plug-in SubjectSource
- Plug-in MapAuthorization
- Plug-in SubjectValidation

Les plug-in valides pour un bean qui est un élément enfant de l'élément backingMapPluginCollection sont inclus dans la liste suivante :

- Plug-in Loader
- Plug-in ObjectTransformer
- Plug-in OptimisticCallback
- Plug-in Evictor
- Plug-in MapEventListener
- Plug-in MapIndex

className
Spécifie le nom de la classe ou du bean Spring à instancier pour créer le plug-in. La classe doit implémenter l'interface de type plug-in. Par exemple, si vous spécifiez `ObjectGridEventListener` comme valeur de l'attribut id, la valeur de l'attribut className doit faire référence à une classe qui implémente l'interface ObjectGridEventListener. (Obligatoire)

```
<bean
(1) id="TransactionCallback" | "ObjectGridEventListener" |"SubjectSource" |
    "MapAuthorization" | "SubjectValidation" | "Loader" | "ObjectTransformer" |
    "OptimisticCallback" | "Evictor" | "MapEventListener" | "MapIndexPlugin"
(2) className="class name" | "(spring)bean name"
/>
```

Dans l'exemple ci-après, le fichier `companyGridBean.xml` permet d'illustrer la manière de configurer des plug-in à l'aide de l'élément bean. Un plug-in ObjectGridEventListener est ajouté à l'ObjectGrid CompanyGrid. L'attribut className de ce bean est la classe com.ibm.websphere.objectgrid.plugins.builtins.TranPropListener. Cette classe implémente l'interface com.ibm.websphere.objectgrid.plugins.ObjectGridEventListener, si nécessaire.

Un plug-in BackingMap est également défini dans le fichier `companyGridBean.xml`. Un plug-in d'expulseur est ajouté à l'instance de mappe de sauvegarde Customer. L'ID bean étant Evictor, l'attribut className doit spécifier une classe qui implémente l'interface com.ibm.websphere.objectgrid.plugins.Evictor. La classe com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor implémente cette interface. La mappe de sauvegarde référence ses plug-in à l'aide de l'attribut pluginCollectionRef.

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

 <objectGrids>
  <objectGrid name="CompanyGrid">
   bean id="ObjectGridEventListener"
    className="com.ibm.websphere.objectgrid.plugins.builtins.TranPropListener"/>
   <backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
  <backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor/>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

L'exemple de code suivant illustre l'approche par programme permettant d'obtenir la même configuration qu'avec le fichier companyGridBean.xml de l'exemple précédent.

```
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);
```

Elément property

L'élément property permet d'ajouter des propriétés aux plug-in. Le nom de la propriété doit correspondre à une méthode set sur la classe référencée par le bean qui la contient.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

name
Indique le nom de la propriété. La valeur affectée à cet attribut doit correspondre à une méthode set sur la classe fournie comme attribut className sur le bean qui la contient. Par exemple, si vous affectez à l'attribut className du bean la valeur com.ibm.MyPlugin et que le nom de la propriété fournie est size, la classe com.ibm.MyPlugin doit contenir une méthode setSize. (Obligatoire)

type
Indique le type de la propriété. Le type est transmis à la méthode set identifiée par l'attribut name. Les valeurs valides sont les primitives Java, les équivalents java.lang et java.lang.String. Les attributs name et type doivent correspondre à une signature de méthode sur l'attribut className du bean. Par exemple, si vous spécifiez le nom size et le type int, une méthode setSize(int) doit exister sur la classe spécifiée comme attribut className pour le bean. (Obligatoire)

value
Indique la valeur de propriété. Cette valeur est convertie dans le type spécifié par l'attribut type, puis utilisée comme paramètre dans l'appel de la méthode set identifiée par les attributs name et type. La valeur de cet attribut n'est pas validée de quelque manière que ce soit. (Obligatoire)

description
Décrit la propriété (facultatif).

```
<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"
/>
```

Dans l'exemple ci-après, le fichier companyGridProperty.xml permet d'illustrer l'ajout d'un élément property à un bean. Dans cet exemple, une propriété dont le nom est maxSize et le type est int est ajoutée à un expulseur. L'expulseur com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor possède une signature de méthode qui correspond à la méthode setMaxSize(int). L'entier 499 est transmis

à la méthode setMaxSize(int) sur la classe 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="The maximum size of the LRU Evictor"/>
   </bean>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

L'exemple de code suivant illustre l'approche par programme permettant d'obtenir la même configuration qu'avec le fichier `companyGridProperty.xml` de l'exemple précédent.

```
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 le fichier XML est utilisé à la place,
// la propriété ajoutée génère l'appel suivant :
lruEvictor.setMaxSize(449);
customerMap.setEvictor(lruEvictor);
```

Elément backingMapPluginsCollections

L'élément backingMapPluginsCollections sert de conteneur à tous les éléments backingMapPluginCollection. Dans le fichier `companyGridProperty.xml` de la section précédente, l'élément backingMapPluginCollections contient un élément backingMapPluginCollection avec l'ID `customerPlugins`.

- nombre d'occurrences : zéro à une
- élément enfant : élément backingMapPluginCollection

Elément backingMapPluginCollection

L'élément backingMapPluginCollection définit les plug-in BackingMap et est identifié par l'attribut **id**. Spécifiez l'attribut pluginCollectionRef pour référencer les plug-in. Si vous configurez plusieurs plug-in BackingMaps de manière similaire, chaque plug-in BackingMap peut faire référence au même élément backingMapPluginCollection.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : élément bean

Attributs

id Identifie la collection de plug-in de mappe de sauvegarde et est référencé par l'attribut pluginCollectionRef de l'élément backingMap. Chaque ID doit être unique. Si la valeur d'un attribut pluginCollectionRef ne correspond pas à l'ID d'un élément backingMapPluginCollection, la validation XML échoue.

Plusieurs éléments backingMap peuvent faire référence à un même élément backingMapPluginCollection. (Obligatoire)

```
<backingMapPluginCollection
(1)   id="id"
/>
```

Dans l'exemple ci-après, le fichier `companyGridCollection.xml` permet d'illustrer la manière d'utiliser l'élément backingMapPluginCollection. Dans ce fichier, la mappe de sauvegarde Customer utilise la collection de plug-in de mappe de sauvegarde customerPlugins pour se configurer avec un expulseur LRU. Les mappes de sauvegarde Item et OrderLine font référence à la collection de plug-in de mappe de sauvegarde collection2. Ces mappes de sauvegarde possède chacune un ensemble d'expulseurs LFU.

```
<?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>
```

L'exemple de code suivant illustre l'approche par programme permettant d'obtenir la même configuration qu'avec le fichier `companyGridCollection.xml` de l'exemple précédent.

```
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");
```

Elément querySchema

L'élément querySchema définit les relations entre les mappes de sauvegarde et identifie le type d'objet dans chaque mappe. Ces informations sont utilisées par la requête d'objet pour convertir les chaînes du langage de requête en appels d'accès à la mappe.

- nombre d'occurrences : zéro à une

- élément enfant : élément mapSchemas, élément relationships

Elément mapSchemas

Chaque élément querySchema possède un élément mapSchemas qui contient un ou plusieurs éléments mapSchema.

- nombre d'occurrences : une
- élément enfant : élément mapSchema

Elément mapSchema

Un élément mapSchema définit le type d'objet stocké dans une mappe de sauvegarde et les instructions d'accès aux données.

- nombre d'occurrences : une à plusieurs
- élément enfant : aucun

Attributs

mapName
Indique le nom de la mappe de sauvegarde à ajouter au schéma. (Obligatoire)

valueClass
Indique le type d'objet stocké dans la portion valeur de la mappe de sauvegarde. (Obligatoire)

primaryKeyField
Indique le nom de l'attribut de clé primaire dans l'attribut valueClass. La clé primaire doit également être stockée dans la portion clé de la mappe de sauvegarde (facultatif).

accessType
Identifie la manière dont le moteur de requête introspecte les données persistantes des instances d'objet valueClass et y accède. Si vous spécifiez la valeur `FIELD`, les champs de classe sont introspectés et ajoutés au schéma. Si la valeur est `PROPERTY`, les attributs associés aux méthodes get et is sont utilisés. La valeur par défaut est `PROPERTY` (facultatif).

```
<mapSchema
(1)   mapName="backingMapName"
(2)   valueClass="com.mycompany.OrderBean"
(3)   primaryKeyField="orderId"
(4)   accessType="PROPERTY" | "FIELD"
/>
```

Dans l'exemple ci-après, le fichier `companyGridQuerySchemaAttr.xml` permet d'illustrer un exemple de configuration mapSchema.

```
<?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>
```

L'exemple de code suivant illustre l'approche par programme permettant d'obtenir la même configuration qu'avec le fichier `companyGridQuerySchemaAttr.xml` de l'exemple précédent.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Définissez le schéma
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);
```

Elément relationships

Chaque élément querySchema possède zéro ou un élément relationships qui contient un ou plusieurs éléments relationship.

- nombre d'occurrences : zéro ou une
- élément enfant : élément relationship

Elément relationship

Un élément relationship définit la relation entre deux mappes de sauvegarde et les attributs de l'attribut valueClass qui associent la relation.

- nombre d'occurrences : une à plusieurs
- élément enfant : aucun

Attributs

source
Indique le nom de la classe de valeur du côté source d'une relation. (Obligatoire)

target
Indique le nom de la classe de valeur du côté cible d'une relation. (Obligatoire)

relationField
Indique le nom de l'attribut dans la classe de valeur source qui fait référence à la cible. (Obligatoire)

invRelationField
Indique le nom de l'attribut dans la classe de valeur cible qui fait référence à la source. Si cet attribut n'est pas spécifié, la relation est unidirectionnelle (facultatif).

```
<mapSchema
(1)   source="com.mycompany.OrderBean"
(2)   target="com.mycompany.CustomerBean"
(3)   relationField="customer"
(4)   invRelationField="orders"
/>
```

Dans l'exemple ci-après, le fichier companyGridQuerySchemaWithRelationshipAttr.xml permet d'illustrer un exemple de configuration mapSchema qui inclut une relation bidirectionnelle.

```
<?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>
```

L'exemple de code suivant illustre l'approche par programme permettant d'obtenir la même configuration qu'avec le fichier companyGridQuerySchemaWithRelationshipAttr.xml de l'exemple précédent.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Définissez le schéma
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);
```

Elément streamQuerySet

L'élément streamQuerySet est l'élément de niveau supérieur de la définition d'un ensemble de requêtes de flux.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : élément stream, élément view

Elément stream

L'élément stream représente un flux vers le moteur de requête de flux. Chaque attribut de l'élément stream correspond à une méthode de l'interface StreamMetadata.

- nombre d'occurrences : une à plusieurs
- élément enfant : élément basic

Attributs

name
Indique le nom du flux. La validation échoue si cet attribut n'est pas spécifié. (Obligatoire)

valueClass
Indique le type de classe de la valeur stockée dans la mappe d'objets de flux. Le type de classe permet de convertir l'objet en événements de flux et de générer une instruction SQL si cette dernière n'est pas fournie. (Obligatoire)

sql
Indique l'instruction SQL du flux. Si cette propriété n'est pas fournie, un SQL de flux est généré en reflétant les attributs ou les méthodes d'accès sur l'attribut valueClass ou en utilisant les attributs de nuplet des métadonnées d'entité (facultatif).

access
Indique le type d'accès aux attributs de la classe de valeur. Si vous spécifiez la valeur`FIELD`, les attributs sont directement extraits des champs à l'aide de la réflexion Java. Sinon, les méthodes d'accès sont utilisées pour lire les attributs. La valeur par défaut est `PROPERTY` (facultatif).

```
<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"
/>
```

Elément view

L'élément view représente une vue de requête de flux. Chaque élément stream correspond à une méthode de l'interface ViewMetadata.

- nombre d'occurrences : une à plusieurs
- élément enfant : élément basic, élément id

Attributs

name
Indique le nom de la vue. La validation échoue si cet attribut n'est pas spécifié. (Obligatoire)

sql
Indique le SQL du flux, qui définit la transformation de la vue. La validation échoue si cet attribut n'est pas spécifié. (Obligatoire)

valueClass
Indique le type de classe de la valeur stockée dans cette vue de la mappe d'objets. Le type de classe permet de convertir les événements de vue dans le format de nuplet approprié compatible avec ce type de classe. Si le type de classe n'est pas fourni, un format par défaut qui suit les définitions de colonne dans le langage SPTSQL (Stream Processing Technology Structured Query Language) est utilisé. Si les métadonnées d'une entité sont définies pour cette mappe de vues, n'utilisez pas l'attribut valueClass (facultatif).

access
Indique le type d'accès aux attributs de la classe de valeur. Si vous spécifiez le type d'accès `FIELD`, les valeurs des colonnes reçoivent directement les valeurs des champs à l'aide de la réflexion Java. Sinon, les méthodes d'accès sont utilisées pour définir les attributs. La valeur par défaut est `PROPERTY` (facultatif).

```
<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"
/>
```

Elément basic

L'élément basic permet de définir un mappage entre le nom d'attribut de la classe de valeur ou les métadonnées d'entité et la colonne définie dans le langage SPTSQL.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

```
<basic
(1)   name="attributeName"
(2)   column="columnName"
/>
```

Elément id

L'élément id est utilisé pour un mappage d'attributs de clé.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

```
<id
(1)   name="idName"
(2)   column="columnName"
/>
```

Dans l'exemple ci-après, le fichier `StreamQueryApp2.xml` permet d'illustrer la configuration des attributs d'un élément streamQuerySet. L'ensemble de requêtes de flux _stockQuoteSQS_ contient un flux et une vue. Ce flux et cette vue définissent son nom, sa classe de valeur, le SQL et le type d'accès. Le flux définit également un élément basic, qui spécifie que l'attribut volume de la classe StockQuote est mappé à la colonne SQL volume de transactions définie dans l'instruction 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>
```

Fichier objectGrid.xsd

Utilisez le schéma XML du descripteur d'ObjectGrid pour configurer WebSphere eXtreme Scale.

Pour les descriptions des éléments et des attributs définis dans le fichier objectGrid.xsd, reportez-vous à la rubrique «Fichier XML du descripteur d'ObjectGrid», à la page 143. Pour plus d'informations sur le fichier objectgrid.xsd de Spring, voir «Fichier XML du descripteur de Spring», à la page 276.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:cc="http://ibm.com/ws/objectgrid/config"
xmlns:dgc="http://ibm.com/ws/objectgrid/config"
elementFormDefault="qualified"
targetNamespace="http://ibm.com/ws/objectgrid/config">

 <xsd:element name="objectGridConfig">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element maxOccurs="1" minOccurs="1" name="objectGrids"
       type="dgc:objectGrids">
     <xsd:unique name="objectGridNameUnique">
      <xsd:selector xpath="dgc:objectGrid"/>
      <xsd:field xpath="@name"/>
     </xsd:unique>
    </xsd:element>
    <xsd:element maxOccurs="1" minOccurs="0" name="backingMapPluginCollections"
       type="dgc:backingMapPluginCollections"/>
   </xsd:sequence>
  </xsd:complexType>

  <xsd:key name="backingMapPluginCollectionId">
   <xsd:selector xpath="dgc:backingMapPluginCollections/dgc:
     backingMapPluginCollection"/>
   <xsd:field xpath="@id"/>
  </xsd:key>

  <xsd:keyref name="pluginCollectionRef" refer="dgc:backingMapPluginCollectionId">
   <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:backingMap"/>
   <xsd:field xpath="@pluginCollectionRef"/>
  </xsd:keyref>

  <xsd:key name="streamName">
   <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:
     streamQuerySet/dgc:stream"/>
   <xsd:field xpath="@name"/>
  </xsd:key>

  <xsd:keyref name="streamRef" refer="dgc:streamName">
   <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:backingMap"/>
   <xsd:field xpath="@streamRef"/>
  </xsd:keyref>

  <xsd:key name="viewName">
   <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:streamQuerySet/dgc:view"/>
   <xsd:field xpath="@name"/>
  </xsd:key>

  <xsd:keyref name="viewRef" refer="dgc:viewName">
   <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:backingMap"/>
   <xsd:field xpath="@viewRef"/>
  </xsd:keyref>
 </xsd:element>

 <xsd:complexType name="objectGrids">
  <xsd:sequence>
   <xsd:element maxOccurs="unbounded" minOccurs="1" name="objectGrid"
      type="dgc:objectGrid">
    <xsd:unique name="backingMapNameUnique">
```

```
    <xsd:selector xpath="dgc:backingMap"/>
    <xsd:field xpath="@name"/>
   </xsd:unique>
   <xsd:unique name="streamQuerySetNameUnique">
    <xsd:selector xpath="dgc:streamQuerySet"/>
    <xsd:field xpath="@name"/>
   </xsd:unique>
  </xsd:element>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="backingMapPluginCollections">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="backingMapPluginCollection"
    type="dgc:backingMapPluginCollection"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="objectGrid">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="bean" type="dgc:bean"/>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="backingMap"
     type="dgc:backingMap"/>
  <xsd:element maxOccurs="1" minOccurs="0" name="querySchema" type="dgc:querySchema"/>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="streamQuerySet"
     type="dgc:streamQuerySet">
   <xsd:unique name="stream">
    <xsd:selector xpath="dgc:stream"/>
    <xsd:field xpath="@name"/>
   </xsd:unique>
   <xsd:unique name="view">
    <xsd:selector xpath="dgc:view"/>
    <xsd:field xpath="@name"/>
   </xsd:unique>
  </xsd:element>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="authorizationMechanism" type="dgc:authorizationMechanism"
   use="optional"/>
 <xsd:attribute name="accessByCreatorOnlyMode" type="dgc:accessByCreatorOnlyMode"
   use="optional"/>
 <xsd:attribute name="securityEnabled" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="txTimeout" type="xsd:int" use="optional"/>
 <xsd:attribute name="permissionCheckPeriod" type="xsd:int" use="optional"/>
 <xsd:attribute name="entityMetadataXMLFile" type="xsd:string" use="optional"/>
 <xsd:attribute name="initialState" type="dgc:initialState" use="optional"/>
</xsd:complexType>


<xsd:complexType name="backingMap">
 <xsd:sequence>
  <xsd:element maxOccurs="1" minOccurs="0" name="timeBasedDBUpdate" type="dgc:
    timeBasedDBUpdate"/>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="readOnly" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="pluginCollectionRef" type="xsd:string" use="optional"/>
 <xsd:attribute name="preloadMode" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="lockStrategy" type="dgc:lockStrategy" use="optional"/>
 <xsd:attribute name="copyMode" type="dgc:copyMode" use="optional"/>
 <xsd:attribute name="valueInterfaceClassName" type="xsd:string" use="optional"/>
 <xsd:attribute name="numberOfBuckets" type="xsd:int" use="optional"/>
 <xsd:attribute name="nullValuesSupported" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="lockTimeout" type="xsd:int" use="optional"/>
 <xsd:attribute name="numberOfLockBuckets" type="xsd:int" use="optional"/>
 <xsd:attribute name="copyKey" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="timeToLive" type="xsd:int" use="optional"/>
 <xsd:attribute name="ttlEvictoryType" type="dgc:ttlEvictorType" use="optional"/>
 <xsd:attribute name="streamRef" type="xsd:string" use="optional"/>
 <xsd:attribute name="viewRef" type="xsd:string" use="optional"/>
 <xsd:attribute name="writeBehind" type="xsd:string" use="optional"/>
 <xsd:attribute name="evictionTriggers" type="xsd:string" use="optional"/>
 <xsd:attribute name="template" type="xsd:boolean" use="optional"/>
</xsd:complexType>

<xsd:complexType name="bean">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="property" type="dgc:property"/>
 </xsd:sequence>
 <xsd:attribute name="className" type="xsd:string" use="required"/>
```

```
 <xsd:attribute name="id" type="dgc:beanId" use="required"/>
</xsd:complexType>

<xsd:complexType name="backingMapPluginCollection">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="bean" type="dgc:bean"/>
 </xsd:sequence>
 <xsd:attribute name="id" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="property">
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="value" type="xsd:string" use="required"/>
 <xsd:attribute name="type" type="dgc:propertyType" use="required"/>
 <xsd:attribute name="description" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:simpleType name="propertyType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="java.lang.Boolean"/>
  <xsd:enumeration value="boolean"/>
  <xsd:enumeration value="java.lang.String"/>
  <xsd:enumeration value="java.lang.Integer"/>
  <xsd:enumeration value="int"/>
  <xsd:enumeration value="java.lang.Double"/>
  <xsd:enumeration value="double"/>
  <xsd:enumeration value="java.lang.Byte"/>
  <xsd:enumeration value="byte"/>
  <xsd:enumeration value="java.lang.Short"/>
  <xsd:enumeration value="short"/>
  <xsd:enumeration value="java.lang.Long"/>
  <xsd:enumeration value="long"/>
  <xsd:enumeration value="java.lang.Float"/>
  <xsd:enumeration value="float"/>
  <xsd:enumeration value="java.lang.Character"/>
  <xsd:enumeration value="char"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="beanId">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="TransactionCallBack"/>
  <xsd:enumeration value="ObjectGridEventListener"/>
  <xsd:enumeration value="SubjectSource"/>
  <xsd:enumeration value="MapAuthorization"/>
  <xsd:enumeration value="SubjectValidation"/>
  <xsd:enumeration value="ObjectGridAuthorization"/>

  <xsd:enumeration value="Loader"/>
  <xsd:enumeration value="ObjectTransformer"/>
  <xsd:enumeration value="OptimisticCallback"/>
  <xsd:enumeration value="Evictor"/>
  <xsd:enumeration value="MapEventListener"/>
  <xsd:enumeration value="MapIndexPlugin"/>

 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="copyMode">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="COPY_ON_READ_AND_COMMIT"/>
  <xsd:enumeration value="COPY_ON_READ"/>
  <xsd:enumeration value="COPY_ON_WRITE"/>
  <xsd:enumeration value="NO_COPY"/>
  <xsd:enumeration value="COPY_TO_BYTES"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="lockStrategy">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="OPTIMISTIC"/>
  <xsd:enumeration value="PESSIMISTIC"/>
  <xsd:enumeration value="NONE"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ttlEvictorType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="CREATION_TIME"/>
  <xsd:enumeration value="LAST_ACCESS_TIME"/>
```

```
      <xsd:enumeration value="LAST_UPDATE_TIME"/>
  <xsd:enumeration value="NONE"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="authorizationMechanism">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="AUTHORIZATION_MECHANISM_JAAS"/>
  <xsd:enumeration value="AUTHORIZATION_MECHANISM_CUSTOM"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="accessByCreatorOnlyMode">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="disabled"/>
  <xsd:enumeration value="complement"/>
  <xsd:enumeration value="supersede"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="streamQuerySet">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="stream" type="dgc:stream">
   <xsd:unique name="streamBasicColumnUnique">
    <xsd:selector xpath="dgc:basic"/>
    <xsd:field xpath="@column"/>
   </xsd:unique>
  </xsd:element>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="view" type="dgc:view">
   <xsd:unique name="viewBasicColumnUnique">
    <xsd:selector xpath="dgc:basic"/>
    <xsd:field xpath="@column"/>
   </xsd:unique>
  </xsd:element>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="viewResultsToListenersOnly" type="xsd:boolean"
    default="false" use="optional"/>
 <xsd:attribute name="deployInPrimaryOnly" type="xsd:boolean" default="true"
    use="optional"/>
</xsd:complexType>

<xsd:complexType name="stream">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="basic" type="dgc:basic"/>
 </xsd:sequence>
 <xsd:attribute name="valueClass" type="xsd:string" use="required"/>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="sql" type="xsd:string" use="optional"/>
 <xsd:attribute name="access" type="cc:accessType" use="optional"/>
</xsd:complexType>

<xsd:complexType name="view">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="id" type="dgc:basic"/>
  <xsd:element element maxOccurs="unbounded" minOccurs="0" name="basic"
    type="dgc:basic"/>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="sql" type="xsd:string" use="optional"/>
 <xsd:attribute name="valueClass" type="xsd:string" use="optional"/>
 <xsd:attribute name="access" type="cc:accessType" use="optional"/>
</xsd:complexType>

<xsd:complexType name="basic">
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="column" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="id">
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="column" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="timeBasedDBUpdate">
 <xsd:attribute name="persistenceUnitName" type="xsd:string" use="optional"/>
 <xsd:attribute name="mode" type="cc:dbUpdateMode" use="optional"/>
 <xsd:attribute name="timestampField" type="xsd:string" use="optional"/>
 <xsd:attribute name="entityClass" type="xsd:string" use="required"/>
 <xsd:attribute name="jpaPropertyFactory" type="xsd:string" use="optional"/>
```

```
</xsd:complexType>

<xsd:simpleType name="dbUpdateMode">
 <xsd:restriction base="xsd:string"/>
  <xsd:enumeration value="INVALIDATE_ONLY"/>
  <xsd:enumeration value="UPDATE_ONLY"/>
  <xsd:enumeration value="INSERT_UPDATE"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="querySchema">
 <xsd:sequence>
  <xsd:element maxOccurs="1" minOccurs="1" name="mapSchemas" type="dgc:mapSchemas">
   <xsd:unique name="mapNameUnique">
    <xsd:selector xpath="dgc:mapSchema"/>
    <xsd:field xpath="@mapName"/>
   </xsd:unique>
  </xsd:element>
  <xsd:element maxOccurs="1" minOccurs="0" name="relationships"
    type="dgc:relationships"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="mapSchemas">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="1" name="mapSchema"
    type="dgc:mapSchema"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="relationships">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="1" name="relationship"
    type="dgc:relationship"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="mapSchema">
 <xsd:attribute name="mapName" type="xsd:string" use="required"/>
 <xsd:attribute name="valueClass" type="xsd:string" use="required"/>
 <xsd:attribute name="primaryKeyField" type="xsd:string" use="optional"/>
 <xsd:attribute name="accessType" type="cc:accessType" use="optional"/>
</xsd:complexType>

<xsd:complexType name="relationship">
 <xsd:attribute name="source" type="xsd:string" use="required"/>
 <xsd:attribute name="target" type="xsd:string" use="required"/>
 <xsd:attribute name="relationField" type="xsd:string" use="required"/>
 <xsd:attribute name="invRelationField" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:simpleType name="accessType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="PROPERTY"/>
  <xsd:enumeration value="FIELD"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="initialState">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="OFFLINE"/>
  <xsd:enumeration value="PRELOAD"/>
  <xsd:enumeration value="ONLINE"/>
 </xsd:restriction>
</xsd:simpleType>

</xsd:schema>
```

Configuration de règles de déploiement

Le fichier XML du descripteur de la règle de déploiement et le fichier XML du descripteur descripteur d'ObjectGrid permettent de gérer une topologie répartie. Le code de la règle de déploiement est écrite en XML dans le fichier fourni au conteneur eXtreme Scale. La règle de déploiement fournit des informations

concernant les mappes, les groupe de mappes, les partitions, les fragments répliques, etc. Elle contrôle également les comportements pour le positionnement des fragments.

Configuration de déploiements répartis

Utilisez le fichier XML du descripteur de la règle de déploiement et le fichier XML du descripteur objectgrid pour gérer votre topologie.

La règle de déploiement est codée en fichier XML fourni au conteneur eXtreme Scale. Le fichier XML spécifie les informations suivantes :

- Les mappes appartenant à chaque groupe de mappes
- Le nombre de partitions
- Le nombre de fragments répliques synchrones et asynchrones

Pour plus d'informations sur le démarrage des serveurs conteneurs, consultez les informations sur le démarrage automatique des conteneurs eXtreme Scale ou le démarrage des processus de conteneur.

La règle de déploiement contrôle également les comportements de positionnement ci-après.

- Le nombre minimum de conteneurs actifs avant positionnement
- Le remplacement automatique des fragments perdus
- Le positionnement de chaque fragment d'une partition sur une autre machine

Pour plus d'informations sur la configuration de la stratégie, consultez les informations sur le fichier XML du descripteur de la règle de déploiement.

Les informations sur les points de contact ne sont pas préconfigurées dans l'environnement dynamique. La règle de déploiement ne contient pas de noms de serveur ou d'informations sur la topologie physique. Tous les fragments d'une grille sont automatiquement placés dans des conteneurs par le service de catalogue. Ce dernier utilise les contraintes définies par la règle de déploiement pour gérer automatiquement le positionnement des fragments. Ce positionnement automatique des fragments permet une configuration aisée pour les grilles de grande taille. Vous pouvez également ajouter des serveurs à votre environnement, si nécessaire.

Restriction : Dans un environnement WebSphere Application Server, une taille de groupe central de plus de 50 membres n'est pas prise en charge.

Un fichier XML de règle de déploiement est transmis à un conteneur eXtreme Scale au démarrage. Une règle de déploiement doit être utilisée avec un fichier XML d'ObjectGrid. Cette règle de déploiement n'est pas requise pour démarrer un conteneur, mais elle est recommandée. La règle de déploiement doit être compatible avec le fichier XML d'ObjectGrid qui lui est associé. Pour chaque élément objectgridDeployment de la règle de déploiement, vous devez inclure un élément objectGrid correspondant dans votre fichier XML d'ObjectGrid. Les mappes de l'élément objectgridDeployment doivent être cohérentes avec les éléments backingMap du XML d'ObjectGrid. Chaque mappe de sauvegarde doit être référencée avec un et un seul élément mapSet.

Dans l'exemple ci-après, le fichier `companyGridDpReplication.xml` doit être associé au fichier `companyGrid.xml` correspondant.

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>
```

Le fichier `companyGridDpReplication.xml` contient un élément mapSet divisé en 11 partitions. Chaque partition doit posséder exactement une réplique synchrone. Le nombre de fragments répliques synchrones est spécifié par les attributs minSyncReplicas et maxSyncReplicas. L'attribut minSyncReplicas ayant la valeur 1, chaque partition de l'élément mapSet doit disposer d'au moins une réplique synchrone disponible pour traiter les transactions d'écriture. L'attribut maxSyncReplicas ayant la valeur 1, chaque partition ne peut dépasser une réplique synchrone. Les partitions de cet élément mapSet ne possèdent pas de fragments répliques asynchrones.

L'attribut numInitialContainers demande au service de catalogue de reporter le positionnement jusqu'à ce que quatre conteneurs soient disponibles pour prendre en charge cette instance ObjectGrid. L'attribut numInitialContainers est ignoré une fois que le nombre de conteneurs spécifié est atteint.

Le fichier `companyGridDpReplication.xml` est un exemple simple, mais une règle de déploiement peut vous permettre de contrôler intégralement votre environnement eXtreme Scale. Consultez les informations sur le fichier XML du descripteur de la règle de déploiement.

Topologie répartie

Les mémoires cache cohérentes réparties permettent d'améliorer les performances, la disponibilité et l'évolutivité du système, que vous pouvez configurer.

WebSphere eXtreme Scale équilibre automatiquement les serveurs. Vous pouvez inclure des serveurs supplémentaires sans redémarrer WebSphere eXtreme Scale. L'ajout de serveurs supplémentaires sans avoir à redémarrer eXtreme Scale permet d'avoir des déploiements simples, mais également des déploiements de grande taille se chiffrant en téraoctets et comptant plusieurs milliers de serveurs.

Cette topologie de déploiement est flexible. A l'aide du service de catalogue, vous pouvez ajouter et supprimer des serveurs afin de mieux utiliser les ressources sans supprimer l'intégralité du cache. Vous pouvez utiliser les commandes

startOgServer et stopOgServer pour démarrer et arrêter les serveurs conteneurs. Ces deux commandes nécessitent de spécifier l'option -catalogServiceEndPoints. Tous les clients d'une topologie répartie communiquent avec le service de catalogue via le protocole IIOP (Internet Interoperability Object Protocol). Tous les clients utilisent l'interface ObjectGrid pour communiquer avec les serveurs.

La fonctionnalité de configuration dynamique de WebSphere eXtreme Scale facilite l'ajout de ressources au système. Les conteneurs hébergent les données et le service de catalogue permet aux clients de communiquer avec la grille de conteneurs. Le service de catalogue transmet les demandes, alloue de l'espace dans les conteneurs hôte et gère l'état et la disponibilité du système global. Les clients se connectent à un service de catalogue, extraient une description de la topologie de serveur conteneur, puis communiquent directement avec chaque serveur. Lorsque la topologie des serveurs change suite à l'ajout de serveurs ou de la défaillance d'autres serveurs, le service de catalogue achemine automatiquement les demandes client au serveur approprié qui héberge les données.

Un service de catalogue existe dans sa propre grille de machines virtuelles Java. Un même serveur de catalogues peut gérer plusieurs serveurs. Vous pouvez démarrer un conteneur dans une machine virtuelle Java ou le charger dans une machine virtuelle Java arbitraire avec d'autres conteneurs de divers serveurs. Un client peut exister dans une machine virtuelle Java et communiquer avec un ou plusieurs serveurs. Un client peut également exister dans la même machine virtuelle Java qu'un conteneur.

Vous pouvez également créer une règle de déploiement à l'aide d'un programme lors de l'imbrication d'un conteneur dans une application ou un processus Java existant. Pour plus d'informations, voir la documentation de l'API DeploymentPolicy d'eXtreme Scale.

Configurer des zones pour le positionnement de fragments répliques

La prise en charge des zones permet la configuration avancée du positionnement de réplique pour plusieurs centres de données. Grâce à cette fonctionnalité, des grilles de milliers de partitions peuvent être facilement gérées à l'aide de plusieurs règles de positionnement facultatives. Un centre de données peut correspondre à plusieurs étages d'un bâtiment, plusieurs bâtiments, plusieurs villes ou d'autres distinctions, selon la configuration des règles de zone.

Flexibilité des zones

Il est possible de placer des fragments dans des zones. Cette fonction permet de mieux contrôler la manière dont eXtreme Scale place les fragments dans une grille. Les machines virtuelles Java qui hébergent un serveur eXtreme Scale peuvent être marquées à l'aide d'un identificateur de zone. Le fichier de déploiement peut désormais inclure une ou plusieurs règles de zone, qui sont associées à un type de fragment. Vous trouverez ci-dessous des exemples et plus de détails.

Les zones de positionnement contrôlent la manière dont eXtreme Scale assigne les fragments primaires et les fragments répliques pour configurer les topologies avancées.

Une machine virtuelle Java peut posséder plusieurs conteneurs mais un seul serveur. Un conteneur peut héberger plusieurs fragments d'un seul objet ObjectGrid.

Cette fonctionnalité permet de s'assurer que les fragments répliques et les fragments primaires sont placés dans différents emplacements ou zones et que leur haute disponibilité est optimale. Généralement, eXtreme Scale ne place pas de fragment principal et de fragement réplique dans les machines virtuelles Java possédant une adresse IP identique. Cette règle simple empêche généralement deux serveurs eXtreme Scale d'être placés sur le même ordinateur physique. Toutefois, vous pouvez avoir besoin d'un mécanisme plus flexible. Par exemple, vous souhaitez utiliser deux châssis lame sur lesquels *segmentés* les fragments primaires et vous voulez que le fragment réplique de chaque fragment primaire soit positionné sur le châssis de l'autre fragment primaire.

Les fragments primaires *à bandes* désignent les fragments primaires placés dans chaque zone. La réplique de chacun de ces fragments primaires est située dans la zone opposée. Par exemple, le fragment primaire 0 se trouve dans la zone A, et lfragment réplique synchronisé 0 dans la zone B. Le fragment primaire 1 se trouve dans la zone B, et le fragment réplique synchronisé 1 dans la zone A.

Dans ce cas, le nom du châssis correspond à celui de la zone. Vous pouvez également nommer les zones en fonction des étages d'un bâtiment et utiliser les zones pour vous assurer que les fragments primaires et les fragments répliques correspondant aux mêmes données se situent à des étages différents. Vous pouvez également utiliser des bâtiments et des centres de données. Des tests effectués sur les centres de données à l'aide de zones ont permis de s'assurer que les données sont répliquées de manière appropriée entre les centres de données. Si vous utilisez HTTP Session Manager pour eXtreme Scale, vous pouvez également utiliser des zones. Cette fonction vous permet de déployer une seule application Web sur les trois centres de données et de vous assurer que les sessions HTTP des utilisateurs sont répliquées dans les centres de données afin que les sessions puissent être récupérées en cas de défaillance d'un centre de données entier.

WebSphere eXtreme Scale prend en compte la nécessité de gérer une grille volumineuse dans plusieurs centres de données. Il est possible de s'assurer que les sauvegardes et les fragments primaires de la même partition sont situés dans des centres de données différents, le cas échéant. Il permet de placer tous les fragments primaires dans le centre de données 1 et tous les fragments répliques dans le centre de données 2. Il peut également permuter de manière circulaire les fragments primaires et les fragments répliques entre les deux centres de données. Les règles sont flexibles de sorte que de nombreux scénarios sont possibles. eXtreme Scale peut également gérer des milliers de serveurs. Cette fonctionnalité, combinée au positionnement automatique en fonction des centres de données, rend ces grilles volumineuses plus économiques d'un point de vue administratif. Les administrateurs peuvent spécifier ce qu'ils veulent faire de manière simple et efficace.

En tant qu'administrateur, utilisez les zones de positionnement pour définir les emplacements des fragments primaires et des fragments répliques. Vous pouvez ainsi configurer des topologies avancées hautes performances et haute disponibilité. Vous pouvez définir une zone dans tout groupement logique de processus eXtreme Scale, comme indiqué ci-dessus : ces zones peuvent correspondre à des emplacements de stations de travail physiques, tels qu'un centre de données, un étage d'un centre de données ou un châssis lame. Vous pouvez segmenter les données dans les zones, afin de bénéficier d'une disponibilité accrue. Vous pouvez également diviser les fragments primaires et les fragments répliques en zones distinctes si un secours automatique est nécessaire.

Association d'un serveur eXtreme Scale à une zone qui n'utilise pas WebSphere Extended Deployment

Si eXtreme Scale est utilisé avec Java Standard Edition ou un serveur d'application qui n'est pas basé sur WebSphere Extended Deployment version 6.1, il est possible d'associer une JVM utilisée comme conteneur de fragments à une zone, si vous utilisez les méthodes suivantes.

Applications utilisant le script startOgServer

Le script startOgServer permet de démarrer une application eXtreme Scale lorsqu'elle n'est pas en cours d'intégration dans un serveur existant. Le paramètre **-zone** permet de spécifier la zone à utiliser pour tous les conteneurs du serveur.

Spécification de la zone lors du démarrage d'un conteneur à l'aide d'API

Association de nœuds WebSphere Extended Deployment à des zones

Si vous utilisez eXtreme Scale avec des applications WebSphere Extended Deployment JEE, vous pouvez optimiser les groupes de nœuds WebSphere Extended Deployment pour placer les serveurs dans des zones spécifiques.

Dans eXtreme Scale, une JVM ne peut être membre que d'une seule zone. Toutefois, WebSphere autorise un nœud à faire partie de plusieurs groupes. Vous pouvez utiliser cette fonctionnalité des zones eXtreme Scale si vous vous assurez que chacun des nœuds se trouve uniquement dans un groupe de nœuds de zone.

Utilisez la syntaxe suivante pour nommer un groupe de nœuds afin de le déclarer en tant que zone : `ReplicationZone<UniqueSuffix>`. Les serveurs exécutés sur un nœud faisant partie d'un tel groupe sont inclus dans la zone spécifiée par le nom du groupe. Vous trouverez ci-dessous la description d'un exemple de topologie.

Tout d'abord, vous devez configurer quatre nœuds : node1, node2, node3 et node4, chaque nœud possédant deux serveurs. Ensuite, vous créez deux groupes de nœuds que vous nommez ReplicationZoneA et ReplicationZoneB. Vous ajoutez node1 et node2 à ReplicationZoneA , et node3 et node4 à ReplicationZoneB.

Lors du démarrage des serveurs de node1 et node2, ils font partie de ReplicationZoneA. De la même manière, les serveurs de node3 et node4 font partie de ReplicationZoneB.

Une machine virtuelle Java membre de grille vérifie l'appartenance aux zones uniquement lors du démarrage. L'ajout d'un nouveau groupe de nœuds ou la modification de l'appartenance a une incidence uniquement sur les machines virtuelles Java démarrées ou redémarrées.

Règles de zone

Une partition eXtreme Scale possède un fragment primaire et aucune réplique ou plus. Pour cet exemple, considérons les conventions d'attribution de nom suivantes pour les fragments. `P` est le fragment primaire ; `S` est une réplique synchrone et `A` une réplique asynchrone. Une règle de zone comporte trois composants :

- un nom de règle
- une liste de zones

- un indicateur inclusif ou exclusif

Le nom de zone d'un conteneur peut être spécifié comme le décrit la documentation de l'API de serveur intégré. Une règle de zone spécifie l'ensemble de zones dans lequel un fragment peut être placé. L'indicateur inclusif indique que le positionnement d'un fragment dans une zone de la liste entraîne le positionnement de tous les autres fragments dans la même liste. Un paramètre exclusif indique que chaque fragment d'une partition est placé dans une zone différente de la liste. Par exemple, si vous utilisez un paramètre exclusif et s'il existe trois fragments (un fragment primaire et deux répliques synchrones), la liste doit alors contenir trois zones.

Chaque fragment peut être associé à une règle de zone. Une règle de zone peut être partagée par deux fragments. Lorsqu'une règle est partagée, l'indicateur inclusif ou exclusif s'applique aux fragments de tout type qui partagent une règle unique.

Exemples

Vous trouverez ci-dessous des exemples illustrant les différents scénarios et la configuration de déploiement permettant d'implémenter ces derniers.

Segmentation des fragments primaires et des fragments répliques dans les zones

Vous disposez de trois châssis lame et souhaitez y répartir les fragments primaires, en plaçant une réplique synchrone dans un châssis différent du fragment primaire. Définissez chaque châssis en tant que zone en les nommant ALPHA, BETA et GAMMA. Exemple de syntaxe XML de déploiement :

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xsi:schemaLocation=
 "http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
    xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="library">
   <mapSet name="ms1" numberOfPartitions="37" minSyncReplicas="1"
    maxSyncReplicas="1" maxAsyncReplicas="0">
   <map ref="book" />
   <zoneMetadata>
    <shardMapping shard="P" zoneRuleRef="stripeZone"/>
    <shardMapping shard="S" zoneRuleRef="stripeZone"/>
    <zoneRule name ="stripeZone" exclusivePlacement="true" >
     <zone name="ALPHA" />
     <zone name="BETA" />
     <zone name="GAMMA" />
    </zoneRule>
   </zoneMetadata>
  </mapSet>
 </objectgridDeployment>
</deploymentPolicy>
```

Cette syntaxe de déploiement XML contient une grille appelée "library" (bibliothèque) qui contient une mappe unique appelée "book". Elle utilise quatre partitions avec une seule réplique synchrone. La clause des métadonnées de zone affiche la définition d'une seule règle de zone et l'association des règles de zone à des fragments. Les fragments primaires et synchrones sont associés à la règle de zone "stripeZone". La règle de zone contient les trois zones et utilise le positionnement exclusif. D'après cette règle, si le fragment primaire de la partition 0 est placé dans ALPHA, le fragment réplique de la partition 0 sera placée dans BETA ou dans GAMMA. De la même manière, les fragments primaires des autres partitions sont placés dans d'autres zones que les fragments répliques.

Réplique asynchrone dans une zone différente de celle du fragment primaire et du fragment réplique synchrone

Dans cet exemple, une connexion avec un temps d'attente élevé existe entre deux bâtiments. Vous souhaitez une haute disponibilité sans perte de données pour tous les scénarios. Toutefois, l'incidence de la réplication synchrone sur les performances entre les bâtiments nécessite un compromis. Vous souhaitez un fragment primaire avec une réplique synchrone dans un bâtiment et une réplique asynchrone dans l'autre bâtiment. Généralement, les défaillances qui se produisent sont des arrêts de JVM ou des blocages de l'ordinateur plutôt que des problèmes à grande échelle. Cette topologie permet d'éviter la perte de données en cas de défaillance normale. La perte d'un bâtiment est suffisamment rare pour qu'une perte de données soit acceptable dans ce cas-là. Vous pouvez créer deux zones, une pour chaque bâtiment. Fichier XML de déploiement :

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

 <objectgridDeployment objectgridName="library">
  <mapSet name="ms1" numberOfPartitions="13" minSyncReplicas="1"
   maxSyncReplicas="1" maxAsyncReplicas="1">
   <map ref="book" />
   <zoneMetadata>
    <shardMapping shard="P" zoneRuleRef="primarySync"/>
    <shardMapping shard="S" zoneRuleRef="primarySync"/>
    <shardMapping shard="A" zoneRuleRef="aysnc"/>
    <zoneRule name ="primarySync" exclusivePlacement="false" >
      <zone name="BldA" />
      <zone name="BldB" />
    </zoneRule>
    <zoneRule name="aysnc" exclusivePlacement="true">
      <zone name="BldA" />
      <zone name="BldB" />
    </zoneRule>
   </zoneMetadata>
  </mapSet>
 </objectgridDeployment>
</deploymentPolicy>
```

Le fragment primaire et le fragment réplique synchrone partagent une règle de zone primaySync avec un paramètre d'indicateur exclusif défini sur "false". Après le positionnement dans une zone du fragment primaire ou de son fragment réplique synchrone, l'autre est placé dans la même zone. La réplique asynchrone utilise une deuxième règle de zone avec les mêmes zones que la règle de zone primarySync mais elle utilise l'attribut **exclusivePlacement** défini sur "true". L'attribut indique qu'un fragment ne peut être placé dans une zone contenant un fragment issu d'une même partition. Par conséquent, le fragment réplique asynchrone n'est pas placé dans la même zone que le fragment primaire ou les fragments répliques synchrones.

Placement de tous les fragments primaires dans une zone et de tous les fragments répliques dans une autre

Dans ce cas, tous les fragments primaires se trouvent dans une zone spécifique et tous les fragments répliques dans une autre zone. Nous obtenons un fragment primaire et une réplique asynchrone unique. Tous les fragments répliques seront placées dans la zone A et les fragments primaires dans la zone B.

```
 <?xml version="1.0" encoding="UTF-8"?>

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

  <objectgridDeployment objectgridName="library">
   <mapSet name="ms1" numberOfPartitions="13" minSyncReplicas="0"
    maxSyncReplicas="0" maxAsyncReplicas="1">
    <map ref="book" />
    <zoneMetadata>
     <shardMapping shard="P" zoneRuleRef="primaryRule"/>
     <shardMapping shard="A" zoneRuleRef="replicaRule"/>
     <zoneRule name ="primaryRule">
      <zone name="A" />
```

```
    </zoneRule>
    <zoneRule name="replicaRule">
     <zone name="B" />
      </zoneRule>
     </zoneMetadata>
    </mapSet>
   </objectgridDeployment>
 </deploymentPolicy>
```

Cet exemple contient deux règles, l'une pour les fragments primaires (P), l'autre pour le fragment réplique (A).

Zones correspondant à des réseaux étendus (WAN)

Vous pouvez souhaiter déployer une instance unique de eXtreme Scale dans plusieurs bâtiments ou centres de données où les interconnexions réseau sont plus lentes. La lenteur accrue des connexions réseau entraîne la réduction de la bande passante et l'augmentation des temps d'attente pour les connexions. Dans ce mode, des partitions réseau sont plus susceptibles de se produire en raison de la congestion du réseau et d'autres facteurs. eXtreme Scale aborde cet environnement difficile de deux manières.

Signal de présence limité entre les zones

Les machines virtuelles Java assemblées en groupes centraux assurent le signal de présence entre elles. Lorsque le service de catalogue organise les machines virtuelles Java en groupes, ces derniers ne s'étendent pas aux zones. Dans ce groupe, un leader transmet les informations d'appartenance au service de catalogue. Ce dernier vérifie les défaillances signalées avant d'entreprendre une action. Pour ce faire, il tente de se connecter aux machines virtuelles Java suspectes. Si le catalogue trouve une fausse détection de défaillance, il n'entreprend pas d'action car la partition de groupe central fonctionnera à nouveau correctement après un court délai.

Le service de catalogue assurera régulièrement le signal de présence des leaders du groupe central à un rythme lent afin de gérer l'isolement du groupe central.

Configuration de la détection des basculements

Le paramètre d'intervalle du signal de présence permet de configurer le laps de temps séparant deux vérifications par le système des serveurs en panne.

Pourquoi et quand exécuter cette tâche

La configuration des basculements varie en fonction du type d'environnement que vous utilisez. Si vous utilisez un environnement autonome, vous pouvez configurer les basculements à l'aide de la ligne de commande. Si vous utilisez un environnement WebSphere Application Server Network Deployment, vous devez les configurer à partir de la console d'administration de WebSphere Application Server Network Deployment.

Procédure

- Configurez les basculements pour les environnements autonomes.

  Vous pouvez configurer les intervalles de signal de présence sur la ligne de commande à l'aide du paramètre **-heartbeat** du fichier script startOgServer.bat | startOgServer.sh. Affectez à ce paramètre l'une des valeurs suivantes :

Tableau 9. Intervalles de signal de présence

Valeur	Action	Description
0	Standard (par défaut)	Les basculements sont généralement détectés dans les 30 secondes.
-1	Elevé	Les basculements sont généralement détectés dans les 5 secondes.
1	Souple	Les basculements sont généralement détectés dans les 180 secondes.

Un intervalle élevé entre les signaux de présence peut être utile si les processus et le réseau sont stables. Si le réseau ou les processus ne sont pas configurés de manière optimale, il peut manquer des signaux de présence, ce qui peut fausser la détection des incidents.

- Configurez les basculements pour les environnements WebSphere Application Server.

  Vous pouvez configurer WebSphere Application Server Network Deployment Version 6.0.2 ou ultérieure pour permettre des basculements très rapides de WebSphere eXtreme Scale. La durée par défaut de pour les incidents matériels est d'environ 200 secondes. Par incident matériel, l'on entend une panne d'ordinateur ou de serveur, une déconnexion de câble réseau ou une erreur du système d'exploitation. Les incidents dus aux pannes de processus ou à des échecs logiciels sont généralement basculés en moins d'une seconde. La détection des incidents logiciels est effectuée lorsque les sockets réseau du processus inactif sont fermés automatiquement par le système d'exploitation du serveur qui héberge le processus.

  Configuration des signaux de présence du groupe central

  Si WebSphere eXtreme Scale est exécuté dans un processus WebSphere Application Server, il hérite des caractéristiques de reprise en ligne des paramètres du groupe central du serveur d'applications. Les sections suivantes décrivent comment configurer les paramètres des signaux de présence du groupe central pour différentes versions de WebSphere Application Server Network Deployment :

  – **Mise à jour des paramètres des groupes centraux de WebSphere Application Server Network Deployment Version 6.x et 7.x :**

    Spécifiez l'intervalle des signaux de présence en secondes sur les versions 6.0 à 6.1.0.12 de WebSphere Application Server ou en millisecondes à partir de la version 6.1.0.13. Vous devez également spécifier le nombre de signaux de présence manqués. Cette valeur indique le nombre maximal de signaux de présence manquants avant qu'une machine virtuelle Java (JVM) ne soit considérée comme défectueuse. Le délai de détection des incidents matériels est approximativement égal au produit de l'intervalle des signaux de présence par le nombre de signaux de présence manqués.

    Ces propriétés sont spécifiées à l'aide des propriétés personnalisées sur le groupe central à l'aide de la console d'administration de WebSphere. Pour des informations de configuration détaillées, voir la rubrique Propriétés personnalisées de groupe central. Ces propriétés doivent être spécifiées pour tous les groupes centraux utilisés par l'application :

    - L'intervalle des signaux de présence est spécifié à l'aide de la propriété personnalisée IBM_CS_FD_PERIOD_SEC pour les secondes ou de la propriété personnalisée IBM_CS_FD_PERIOD_MILLIS pour les millisecondes (nécessite la version 6.1.0.13 ou une version ultérieure).
    - Le nombre de signaux de présence manqués est spécifié à l'aide de la propriété personnalisée IBM_CS_FD_CONSECUTIVE_MISSED.

La valeur par défaut de la propriété IBM_CS_FD_PERIOD_SEC est de 20 et celle de la propriété IBM_CS_FD_CONSECUTIVE_MISSED, de 10. Si la propriété IBM_CS_FD_PERIOD_MILLIS est spécifiée, elle remplace les propriétés personnalisées IBM_CS_FD_PERIOD_SEC définies. Les valeurs de ces propriétés correspondent à des entiers.

Utilisez les paramètres suivants pour spécifier un délai de détection des incidents de 1500 ms pour les serveurs WebSphere Application Server Network Deployment Version 6.x :

- Spécifiez `IBM_CS_FD_PERIOD_MILLIS = 750` (WebSphere Application Server Network Deployment V6.1.0.13 et versions ultérieures)
- Spécifiez `IBM_CS_FD_CONSECUTIVE_MISSED = 2`

– **Mise à jour des paramètres des groupes centraux de WebSphere Application Server Network Deployment Version 7.0**

WebSphere Application Server Network Deployment Version 7.0 fournit deux paramètres de groupe central qui peuvent être ajustés pour augmenter ou réduire le délai de détection des incidents :

- **Période de transmission du signal de présence.** La valeur par défaut est de 30000 millisecondes.
- **Période d'expiration du signal de présence.** La valeur par défaut est de 180000 millisecondes.

Pour plus de détails sur la manière de modifier ces paramètres, reportez-vous à la rubrique Paramètres de reconnaissance et de détection des incidents du Centre de documentation de WebSphere Application Server Network Deployment .

Utilisez les paramètres suivants pour spécifier un délai de détection des incidents de 1500 ms pour les serveurs WebSphere Application Server Network Deployment Version 7 :

- Spécifiez une période de transmission du signal de présence de 750 millisecondes.
- Spécifiez une période d'expiration du signal de présence de 1500 millisecondes.

Que faire ensuite

Lorsque vous modifiez ces paramètres pour réduire les délais de basculement, certains points d'optimisation du système sont à prendre en compte. Tout d'abord, Java n'est pas un environnement en temps réel. Des unités d'exécution peuvent être retardées si la JVM connaît des délais de récupération de place importants. Les unités d'exécution risquent également d'être retardées si la charge de la machine qui héberge la JVM est considérable (à cause de la JVM elle-même ou d'autres processus exécutés sur cette machine). Si les unités d'exécution sont retardées, les signaux de présence risquent de ne pas être envoyés à temps. Au pire, ils risquent d'être retardés du délai requis pour la reprise en ligne. Si des unités d'exécution sont retardées, des incidents sont détectés à tort. Le système doit être optimisé et dimensionné de sorte à éviter la détection de faux incidents en production. Il est recommandé pour cela de tester la charge de manière adéquate.

Remarque : La version actuelle d'eXtreme Scale prend en charge WebSphere Real Time.

Fichier XML du descripteur de la règle de déploiement

Pour configurer une règle de déploiement, utilisez un fichier XML de descripteur de règle de déploiement.

Dans les sections ci-après, les éléments et les attributs du fichier XML du descripteur de règle de déploiement sont définis. Voir la rubrique «Fichier `deploymentPolicy.xsd`», à la page 180 pour le schéma XML de la règle de déploiement correspondante.

Eléments dans le fichier deploymentPolicy.xml

```
(1) <deploymentPolicy>
(2)  <objectgridDeployment objectGridName="blah">
(3)   <mapSet
(4)    name="mapSetName"
(5)    numberOfPartitions="numberOfPartitions"
(6)    minSyncReplicas="minimumNumber"
(7)     maxSyncReplicas="maximumNumber"
(8)    maxAsyncReplicas="maximumNumber"
(9)    replicaReadEnable="true|false"
(10)    numInitialContainers="numberOfInitialContainersBeforePlacement"
(11)    autoReplaceLostShards="true|false"
(12)    developmentMode="true|false"
(13)    placementStrategy="FIXED_PARTITION|PER_CONTAINER">
(14)    <map ref="backingMapReference" />
(15)   </mapSet>
(16)   <zoneMetadata>
(17)    <shardMapping
(18)      shard="shardName"
(19)      zoneRuleRef="zoneRuleRefName" />
(20)    <zoneRule
(21)      name="zoneRuleName"
(22)      exclusivePlacement="true|false" >
(23)      <zone name="ALPHA" />
(24)      <zone name="BETA" />
(25)      <zone name="GAMMA" />
(26)    </zoneRule>
(27)   </zoneMetadata>
(28)  </objectgridDeployment>
</deploymentPolicy>
```

deploymentPolicy element (line 1)

L'élément deploymentPolicy correspond à l'élément de niveau supérieur du fichier XML de la règle de déploiement. Cet élément configure l'espace de noms du fichier et l'emplacement du schéma. Le schéma est défini dans le fichier `deploymentPolicy.xsd`.

- **nombre d'occurrences :** une
- **élément enfant :** objectgridDeployment

Elément objectgridDeployment (ligne 2)

L'élément objectgridDeployment permet de référencer une instance ObjectGrid à partir du fichier XML ObjectGrid. Dans l'élément objectgridDeployment, vous pouvez diviser vos mappes en groupes de mappes.

- **nombre d'occurrences :** une ou plusieurs
- **élément enfant :** mapSet

Attributs

objectgridName
Indique le nom de l'instance ObjectGrid à déployer. Cet attribut fait référence à un élément objectGrid défini dans le fichier XML ObjectGrid (obligatoire).

Par exemple, l'attribut objectgridName possède la valeur `CompanyGrid` dans le fichier companyGridDpReplication.xml. L'attribut objectgridName fait référence à la valeur `CompanyGrid` définie dans le fichier companyGrid.xml. Consultez la rubrique «Fichier XML du descripteur d'ObjectGrid», à la page 143 en association avec le fichier des règles de déploiement de chaque instance ObjectGrid.

Elément mapSet (ligne 3)

L'élément mapSet permet de reqrouper les mappes ensemble. Les mappes d'un élément mapSet sont partitionnées et répliquées de la même manière. Chaque mappe ne doit appartenir qu'à un élément mapSet.

- **nombre d'occurrences :** une ou plusieurs
- **élément enfant :** map

Attributs

name
Indique le nom du groupe de mappes. Cet attribut doit être unique dans l'élément objectgridDeployment (obligatoire).

numberOfPartitions
Indique le nombre de partitions de l'élément mapSet. La valeur par défaut est 1. Ce nombre doit être approprié pour le nombre de conteneurs qui hébergent les partitions (facultatif).

minSyncReplicas
Indique le nombre minimal de fragments répliques synchrones de chaque partition du mapSet. La valeur par défaut est 0. Les fragments ne sont pas placés tant que le domaine ne peut pas prendre en charge le nombre minimal de fragments répliques synchrones. Pour prendre en charge la valeur minSyncReplicas, vous avez besoin d'un nombre de conteneurs égal à la valeur de minSyncReplicas plus un. Si le nombre de fragments répliques synchrones est inférieur à la valeur de minSyncReplicas, les transactions d'écrire ne sont plus autorisées pour cette partition (facultatif).

maxSyncReplicas
Indique le nombre maximal de fragments répliques synchrones de chaque partition du mapSet. La valeur par défaut est 0. Aucune autre réplique synchrone n'est placée pour une partition une fois qu'un domaine a atteint ce nombre de fragments répliques synchrones pour cette partition spécifique. L'ajout de conteneurs qui peuvent prendre en charge cet ObjectGrid peut augmenter le nombre de fragments répliques synchrones si votre valeur maxSyncReplicas n'est pas encore atteinte (facultatif).

maxAsyncReplicas
Indique le nombre maximal de fragments répliques asynchrones de chaque partition du mapSet. La valeur par défaut est 0. Une fois que le fragment primaire et tous les fragments répliques synchrones ont été placés pour une partition, les fragments répliques asynchrones sont placés jusqu'à ce que la valeur maxAsyncReplicas soit atteinte (facultatif).

replicaReadEnabled
Si cet attribut est défini sur `true`, les demandes de lecture sont réparties entre le fragment primaire d'une partition et ses fragments répliques. Si l'attribut replicaReadEnabled est défini sur `false`, les demandes de lecture ne sont acheminées que vers le fragment primaire. La valeur par défaut est false (facultatif).

numInitialContainers
Indique le nombre de conteneurs eXtreme Scale requis avant le positionnement initial des fragments de cet élément mapSet. La valeur par défaut est 1. Cet attribut peut permettre d'économiser la bande passante des processus et du réseau lorsqu'une instance ObjectGrid est mise en ligne (facultatif).

Le démarrage d'un conteneur eXtreme Scale envoie un événement au service de catalogue. La première fois que le nombre de conteneurs actifs est égal à la valeur numInitialContainers d'un élément mapSet, le service de catalogue place les fragments du groupe de mappes, à condition que la valeur minSyncReplicas puisse également être atteinte. Une fois que la valeur numInitialContainers a été atteinte, chaque événement démarré par un conteneur peut déclencher un rééquilibrage des fragments non positionnés et positionnés précédemment. Si vous connaissez à peu près le nombre de conteneurs que vous allez démarrer pour cet élément mapSet, vous pouvez spécifier une valeur numInitialContainers proche de ce nombre pour éviter le rééquilibrage après chaque démarrage de conteneur. Le positionnement n'est effectué que si vous atteignez la valeur numInitialContainers spécifiée dans l'élément mapSet.

autoReplaceLostShards
Indique si les fragments perdus sont placés dans d'autres conteneurs. La valeur par défaut est true. Si un conteneur est arrêté ou échoue, les fragments exécutés sur le conteneur sont perdus. Lors de la perte d'un fragment primaire, l'un de ses fragments réplique est promu comme fragment primaire pour la partition correspondante. En raison de cette promotion, l'un des serveur secondaire est perdu. Si vous souhaitez que les fragments perdus ne soient pas placés, affectez à l'attribut autoReplaceLostShards la valeur `false`. Ce paramètre n'affecte pas la chaîne de promotion, mais seulement le remplacement du dernier fragment de la chaîne (facultatif).

developmentMode
Avec cet attribut, vous pouvez influencer le positionnement d'un fragment par rapport à ses fragments homologues. La valeur par défaut est true. Si l'attribut developmentMode a la valeur `false`, deux fragments d'une même partition ne peuvent pas être placés sur un même ordinateur. Si l'attribut developmentMode a la valeur `true`, les fragments d'une même partition peuvent être placés sur une même machine. Dans ces deux cas, les fragments d'une même partition ne sont jamais placés dans un même conteneur (facultatif).

placementStrategy
Il existe deux stratégies de positionnement. La stratégie par défaut est `FIXED_PARTITION`, dans laquelle le nombre de fragments primaires placés entre les conteneurs disponibles est égal au nombre de partitions définies auquel est ajouté le nombre de fragments répliques. L'autre stratégie est `PER_CONTAINER`, dans laquelle le nombre de fragments primaires placés sur chaque conteneur est égal au nombre de partitions définies, avec un nombre identique de fragments répliques placés sur les autres conteneurs (facultatif).

Elément map (ligne 14)

Chaque mappe d'un élément mapSet fait référence à l'un des éléments backingMap définis dans le fichier XML ObjectGrid correspondant. Chaque mappe d'un environnement eXtreme Scale réparti ne peut appartenir qu'à un élément mapSet.

- **nombre d'occurrences :** une ou plusieurs
- **élément enfant :** aucun

Attributs

ref
Fournit une référence à un élément backingMap dans le fichier XML ObjectGrid. Chaque mappe d'un élément mapSet doit faire référence à un

élément backingMap du fichier XML ObjectGrid. La valeur affectée à l'attribut ref doit correspondre à l'attribut name de l'un des éléments backingMap du fichier XML d'ObjectGrid, comme dans le fragment de code ci-après (obligatoire).

Elément zoneMetadata (ligne 16)

Il est possible de placer des fragments dans des zones. Cette fonction permet de mieux contrôler la manière dont eXtreme Scale positionne les fragments dans la grille. Les machines virtuelles Java qui hébergent un serveur eXtreme Scale peuvent être marquées à l'aide d'un identificateur de zone. Le fichier de déploiement peut inclure une ou plusieurs règles de zone, lesquelles sont associées à un type de fragment. Pour plus d'informations, voir «Configurer des zones pour le positionnement de fragments répliques», à la page 167.

- **nombre d'occurrences :** zéro ou une
- **éléments enfants :**
  - shardMapping
  - zoneRule

Attributs : aucun

élément shardMapping (ligne 17)

Chaque fragment peut être associé à une règle de zone. Une règle de zone peut être partagée par deux fragments. Lorsqu'une règle est partagée, l'indicateur inclusif ou exclusif s'applique aux fragments de tout type qui partagent une règle unique.

- **nombre d'occurrences :** zéro ou une
- **éléments enfants :** aucun

Attributs

shard
Spécifiez le nom d'un fragment auquel associer la règle de zone (obligatoire).

zoneRuleRef
Spécifiez le nom de la règle de zone à laquelle associer le fragment (facultatif).

élément zoneRule (ligne 20)

Une règle de zone spécifie l'ensemble des zones possibles dans lequel un fragment peut être positionné.

- **nombre d'occurrences :** une ou plusieurs
- **éléments enfants :** zone

Attributs

name
Spécifiez le nom de la règle de zone que vous avez précédemment définie comme zoneRuleRef dans un élément shardMapping (obligatoire).

exclusivePlacement
Un paramètre exclusif indique que chaque fragment d'une partition est placé dans une zone différente de la liste. Le paramètre inclusif indique que le positionnement d'un fragment dans une zone de la liste entraîne le positionnement dans cette zone de tous les autres fragments. Par exemple, si

vous utilisez un paramètre exclusif et qu'il existe trois fragments (un fragment primaire et deux fragments répliques synchrones), la liste devra contenir trois zones (facultatif).

Elément zone (lignes 23 à 25)

Supposons que vous disposiez de trois châssis lame et que vous souhaitiez y répartir les fragments principaux, en positionnant un fragment réplique synchrone dans un autre châssis que le fragment principal. Vous pouvez définir chaque châssis comme zone en donnant à ces zones des noms correspondant à ceux des châssis : ALPHA, BETA et GAMMA.

- **nombre d'occurrences :** une ou plusieurs
- **éléments enfants :** aucun

Attributs

name
Spécifiez le nom de la zone à laquelle vous voulez appliquer la règle (obligatoire).

Exemple

Dans l'exemple ci-après, l'élément mapSet est utilisé pour configurer une règle de déploiement. La valeur est `mapSet1` et elle est divisée en 10 partitions. Pour chacune de ces partitions, une ou deux répliques synchrones doivent être disponibles. Chaque partition possède également une réplique asynchrone si l'environnement peut la prendre en charge. Tous les fragments répliques synchrones sont placés avant les éventuels fragments répliques asynchrones. En outre, le service de catalogue ne tente pas de placer les fragments de l'élément mapSet1 tant que le domaine ne prend pas en charge la valeur minSyncReplicas. La prise en charge de la valeur minSyncReplicas requiert plusieurs conteneurs : un pour le fragment primaire et deux pour le fragment réplique synchrone.

```
<?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>
```

Deux conteneurs seulement sont requis pour satisfaire les paramètres de réplication, mais l'attribut numInitialContainers requiert 10 conteneurs disponibles pour que le service de catalogue puisse positionner l'un des fragments dans cet élément mapSet. Une fois que le domaine dispose de 10 conteneurs qui peuvent prendre en charge l'ObjectGrid CompanyGrid, tous les fragments de l'élément mapSet1 sont positionnés.

L'attribut autoReplaceLostShards possédant la valeur true, tout fragment de cet élément mapSet qui est perdu suite à un incident de conteneur est automatiquement repositionné sur un autre conteneur, à condition qu'un conteneur soit disponible pour héberger le fragment perdu. Les fragments d'une même partition ne peuvent pas être positionnés sur la même machine que l'élément

mapSet1 car l'attribut developmentMode possède la valeur `false`. Les demandes en lecture seule sont réparties entre le fragment primaire et ses fragments répliques pour chaque partition car la valeur de replicaReadEnabled est `true`.

Le fichier companyGridDpMapSetAttr.xml utilise l'attribut ref sur la mappe pour faire référence à chacun des éléments backingMap du fichier companyGrid.xml.

Fichier `deploymentPolicy.xsd`

Utilisez le schéma XML de la règle de déploiement pour créer un fichier XML de descripteur de déploiement.

Pour les descriptions des éléments et des attributs définis dans le fichier `deploymentPolicy.xsd`, reportez-vous à la rubrique «Fichier XML du descripteur de la règle de déploiement», à la page 174.

```
<?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>
```

Configuration de serveurs catalogues et de serveurs conteneur

Utilisez un fichier de propriétés pour configurer des serveur de catalogue et des serveurs conteneurs. WebSphere eXtreme connaît deux types de serveurs : les serveurs de catalogue et les serveurs conteneurs. Les serveurs de catalogues contrôlent le positionnement des fragments et détectent et surveillent les serveurs conteneurs. Ensemble, plusieurs serveurs de catalogues constituent le service de catalogue. Les serveurs conteneurs sont les machines virtuelles Java qui stockent les données d'application de la grille.

Configuration de topologies de réplication multi-maître

La réplication asynchrone multi-maître permet d'utiliser des liens pour interconnecter un ensemble de domaines et de faire synchroniser ensuite ces derniers par eXtreme Scale grâce, précisément, à la réplication via ces liens. Comme c'est vous qui définissez les liens entre les domaines, vous pouvez élaborer quasiment n'importe quelle topologie. Vous définissez les liens dans les fichiers de

propriétés des serveurs de catalogue de chaque domaine. Vous pouvez également définir les liens au moment de l'exécution à l'aide de programmes JMX ou de l'utilitaire xsadmin de ligne de commande. Et, lorsque vous créez des liens, l'ensemble des liens actuels pour un domaine est stocké dans le service de catalogue, ce qui vous permet d'ajouter et de supprimer des liens sans avoir à redémarrer le domaine (la grille).

Avant de commencer

Topologies de réplication de grilles multi-maîtres (PA) vous initie aux caractéristiques des diverses topologies de réplication multi-maître. La procédure qui suit décrit la configuration de liens variés entre des domaines afin de constituer une topologie, n'importe quelle topologie. A la suite de la procédure, plusieurs exemples illustreront comment configurer des topologies spécifiques, une formation en étoile, par exemple.

Procédure

1. Définissez des liens dans les fichiers de propriétés du serveur de catalogue de chacun des domaines de la topologie à des fins d'amorçage.

   Le fichier de propriétés est automatiquement détecté si vous le nommez `objectGridServer.properties` (respect obligatoire des majuscules et des minuscules sur certains systèmes) et que vous le placez dans le chemin d'accès aux classes utilisé au démarrage d'une instance de service de catalogue. Vous pouvez également passer par la ligne de commande pour spécifier son emplacement au script `startOgServer.bat|sh`, à l'aide du paramètre *-serverProps*.

   Veillez à bien respecter les majuscules et les minuscules des noms de propriétés lorsque vous modifiez le fichier des propriétés.

   Nom du domaine local
   Spécifiez le nom de "ce" domaine, comme le domaine A, par exemple :

   ```
   domainName=A
   ```

   Liste facultative des noms des domaines externes

   Spécifiez les noms des "autres" domaines de la topologie de réplication multi-maître, comme le domaine B, par exemple :

   ```
   foreignDomains=B
   ```

   Liste facultative des points de contact des noms des domaines externes
   Spécifie les informations de connexion pour les services de catalogue des domaines externes comme le domaine B :

   Par exemple :

   ```
   B.endPoints=hostB1:2809, hostB2:2809
   ```

   Si un domaine externe comporte plusieurs services de catalogue, spécifiez-les tous.

2. Utilisez xsadmin ou la programmation JMX pour ajouter ou supprimer des liens au moment de l'exécution.

   Les liens d'un domaine sont conservés dans le service de catalogue dans la mémoire répliquée. Cet ensemble de liens peut être modifié à tout moment par l'administrateur sans nécessiter pour autant un redémarrage de ce domaine ou des autres domaines. L'utilitaire xsadmin de ligne de commande inclut plusieurs options pour l'utilisation des liens.

L'utilitaire xsadmin se connecte à un service de catalogue et, de ce fait, à un seul domaine. En conséquence de quoi, xsadmin peut être utilisé pour créer et supprimer des liens entre le domaine auquel il se connecte et n'importe quel autre domaine.

La ligne de commande permet de créer un lien, par exemple :

```
xsadmin –ch host –p 1099 –establishLink dname fdHostA:2809,fdHostB:2809
```

Cette commande établit un nouveau lien entre le domaine "local" et le domaine externe nommé "dname" dont le service de catalogue s'exécute sur fdHostA:2809 et sur fdHostB:2809. Le domaine local a une machine virtuelle Java de service de catalogue avec une adresse JMX host:1099. N'omettez pas de spécifier tous les points de contact du domaine externe, faute de quoi la connectivité de la tolérance aux pannes ne serait pas possible. Il n'est pas recommandé d'utiliser une seule paire hôte:port pour le service de catalogue du domaine externe.

La machine virtuelle Java de service de catalogue local spécifiée par xsadmin à l'aide de –ch et de –p n'a pas d'importance en elle-même. Toutes les machines virtuelles Java de catalogue fonctionneront. Si le catalogue est hébergé sur un gestionnaire de déploiement WebSphere Application Server, le port est habituellement le 9809.

Les ports spécifiés pour le domaine externe NE SONT PAS des ports JMX. Ce sont les ports que l'on utilise d'ordinaire pour les clients eXtreme Scale.

Une fois que la commande d'ajout de nouveau lien a été émise, le service de catalogue donne instruction à tous les conteneurs qu'il gère de commencer à se répliquer vers le domaine externe. Un lien n'est pas nécessaire des deux côtés. Il suffit d'en créer un sur l'une des deux extrémités.

La ligne de commande permet également de supprimer un lien, par exemple :

```
xsadmin –ch host –p 1099 –dismissLink dname
```

Cette commande se connecte au service de catalogue d'un domaine et lui donne instruction d'arrêter la réplication vers un domaine spécifique. Un lien n'a pas besoin d'être supprimé des deux côtés, un seul suffit.

Exemple

Supposons que vous vouliez configurer une formation à deux domaines, les domaines A et B.

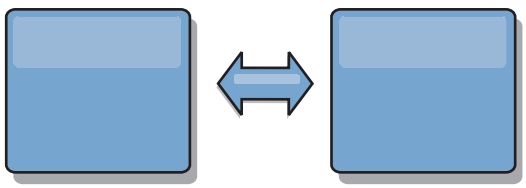


Figure 8. Lien entre des domaines

Voici le fichier de propriétés du serveur de catalogue sur le domaine A :

```
domainName=A
foreignDomains=B
B.endPoints=hostB1:2809, hostB2:2809
```

Voici le fichier de propriétés du serveur de catalogue sur le domaine B. Vous remarquerez la ressemblance entre les deux fichiers de propriétés.

```
domainName=B
foreignDomains=A
B.endPoints=hostB1:2809, hostB2:2809
```

Après le démarrage des deux domaines, toutes les grilles présentant les caractéristiques suivantes seront répliquées entre ces deux domaines :

- disposer d'un service de catalogue privé avec un nom de domaine exclusif
- avoir le même nom de grille que les autres grilles du domaine
- avoir le même nombre de partitions que les autres grilles du domaine
- être une grille FIXED_PARTITION (les grilles PER_CONTAINER ne peuvent être répliquées)
- avoir le même nombre de partitions (sans forcément pour autant avoir le même nombre et le même type de fragments répliques)
- avoir les mêmes types de données à répliquer que les autres grilles du domaine
- avoir le même nom de groupes de mappes (mapsets), le même nom de mappes et les mêmes modèles de mappes dynamiques que les autres grilles du domaine

Il est à noter que les règles de réplication des domaines seront ignorées.

L'exemple qui précède montre comment configurer chaque domaine pour qu'il ait un lien vers l'autre domaine, mais, en fait, il suffit de définir un lien dans une seule direction. C'est particulièrement utile lorsqu'on a affaire à des topologies en étoile, la configuration s'en trouve considérablement simplifiée. Le fichier de propriétés du concentrateur ne nécessite pas d'être modifié au fur et à mesure que des noeuds sont ajoutés à la topologie et il suffit que le fichier de chacun de ces noeuds comprenne des informations relatives au concentrateur. De la même manière, dans une topologie en anneau, il suffit que chacun des domaines ait un lien avec le domaine qui le précède et avec celui qui le suit dans l'anneau.

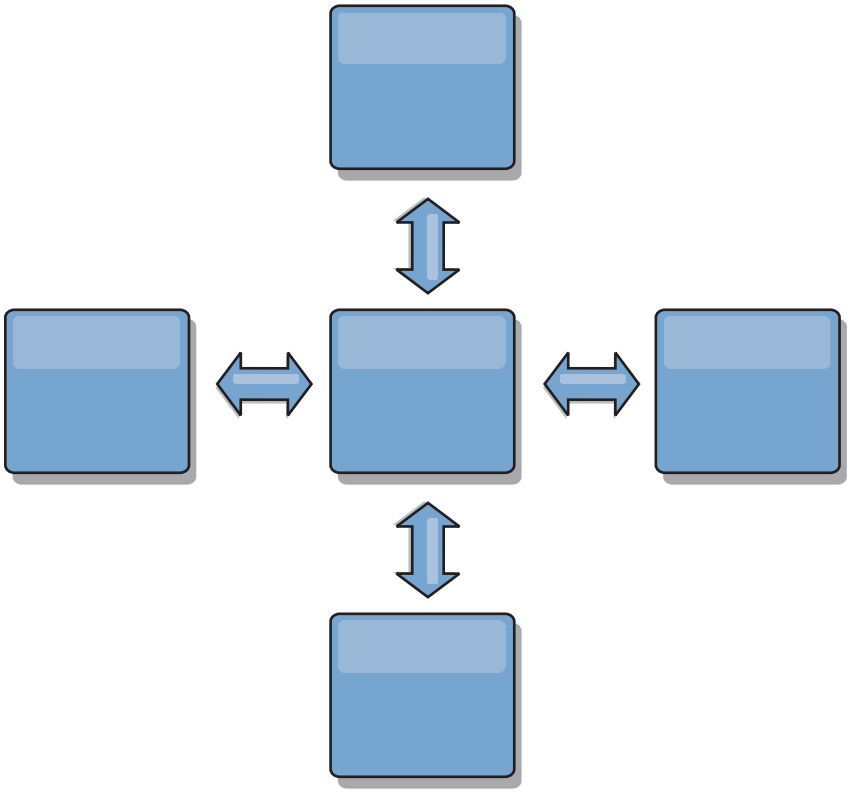


Figure 9. Topologie en étoile

Les quatre domaines périphériques (domaines A, B, C et D) auront des fichiers de propriétés de serveur de catalogue semblables aux exemples qui suivent.

```
domainName=Hub
```

Le premier ordinateur périphérique aura les propriétés suivantes :

```
domainName=A
foreignDomains=Hub
Hub.endPoints=hostH1:2809, hostH2:2809
```

Le deuxième aura les propriétés suivantes :

```
domainName=B
foreignDomains=Hub
Hub.endPoints=hostH1:2809, hostH2:2809
```

Le troisième aura les propriétés suivantes :

```
domainName=C
foreignDomains=Hub
Hub.endPoints=hostH1:2809, hostH2:2809
```

Le quatrième aura les propriétés suivantes :

```
domainName=D
foreignDomains=Hub
Hub.endPoints=hostH1:2809, hostH2:2809
```

Fichier de propriétés du serveur

Le fichier de propriétés du serveur contient plusieurs propriétés qui définissent différents paramètres pour votre serveur, tels que les paramètres de trace, la

consignation et la configuration de la sécurité. Le fichier de propriétés du serveur est utilisé par le service de catalogue et les serveurs conteneurs.

Exemples de fichier de propriétés du serveur

Vous pouvez utiliser le fichier `sampleServer.properties` qui se trouve dans le répertoire *racine_extremescale*`/properties` pour créer votre fichier de propriétés.

Spécification d'un fichier de propriétés du serveur

Vous pouvez spécifier le fichier de propriétés du serveur de l'une des manières ci-après. La spécification d'un paramètre en utilisant l'un des éléments plus loin dans la liste remplace le paramètre précédent. Par exemple, si vous spécifiez une valeur de propriété système pour le fichier de propriétés du serveur, les propriétés de ce fichier remplacent les valeurs du fichier `objectGridServer.properties` qui se trouvent dans le chemin d'accès aux classes.

1. Comme fichier nommé correctement dans le chemin d'accès aux classes. Si vous placez ce fichier nommé correctement dans le répertoire actuel, le fichier est introuvable s'il ne se trouve pas dans le chemin d'accès aux classes. Le nom utilisé est le suivant :

   ```
   objectGridServer.properties
   ```

2. Comme propriété système dans une configuration autonome ou WebSphere Application Server qui spécifie un fichier dans le répertoire actuel du système. Le fichier ne peut pas se trouver dans le chemin d'accès aux classes :

   ```
   -Dobjectgrid.server.props=nom_fichier
   ```

3. Comme paramètre lorsque vous exécutez la commande startOgServer. Vous pouvez remplacer ces propriétés manuellement pour spécifier un fichier dans le répertoire actuel du système :

   ```
   -serverProps nom_fichier
   ```

4. Comme une substitution à l'aide d'un programme, à l'aide des méthodes ServerFactory.getServerProperties et ServerFactory.getCatalogServerProperties. Les données de l'objet sont alimentées avec celles des fichiers de propriétés.

Propriétés du serveur

Propriétés générales

workingDirectory
Indique l'emplacement dans lequel la sortie du serveur conteneur est enregistrée. Si cette valeur n'est pas spécifiée, la sortie est enregistrée dans un répertoire `log` du répertoire actuel. Cette propriété s'applique à la fois au serveur conteneur et au service de catalogue.

Valeur par défaut : aucune

minThreads
Indique le nombre minimal d'unités d'exécution utilisées par le pool d'unités d'exécution interne dans l'environnement d'exécution pour les expulseurs pré-intégrés et pour les opérations DataGrid.

Valeur par défaut : 10

maxThreads
Indique le nombre maximal d'unités d'exécution utilisées par le pool d'unités d'exécution interne dans l'environnement d'exécution pour les expulseurs pré-intégrés et pour les opérations DataGrid.

Valeur par défaut : 50

traceSpec
Active la trace et la chaîne de spécification de trace du serveur conteneur. La trace est désactivée par défaut. Cette propriété s'applique à la fois au serveur conteneur et au service de catalogue.

Valeur par défaut : `*=all=disabled`

traceFile
Indique le nom du fichier dans lequel les informations de trace seront consignées. Cette propriété s'applique à la fois au serveur conteneur et au service de catalogue.

systemStreamToFileEnabled
Permet au conteneur de consigner les sorties SystemOut, SystemErr et de trace dans un fichier. Si cette propriété a la valeur `false`, la sortie n'est pas consignée dans un fichier, mais affichée dans la console.

Valeur par défaut : `true`

enableMBeans
Actives les beans gérés par conteneur ObjectGrid (MBean). Cette propriété s'applique à la fois au serveur conteneur et au service de catalogue.

Valeur par défaut : `true`

serverName
Définit le nom de serveur utilisé pour identifier le serveur. Cette propriété s'applique à la fois au serveur conteneur et au service de catalogue.

zoneName
Définit le nom de la zone à laquelle le serveur appartient. Cette propriété s'applique à la fois au serveur conteneur et au service de catalogue.

haManagerPort
Synonyme de port homologue. Indique le numéro de port utilisé par le gestionnaire de haute disponibilité. Si cette propriété n'est pas définie, le service de catalogue génère automatiquement un port disponible. Cette propriété s'applique à la fois au serveur conteneur et au service de catalogue.

listenerHost
Indique le nom d'hôte auquel l'ORB (Object Request Broker) doit être associé. Cette propriété s'applique à la fois au serveur conteneur et au service de catalogue.

Si votre configuration implique la présence de plusieurs cartes réseau, définissez l'hôte et le port d'écoute pour faire connaître à l'ORB de la machine virtuelle Java l'adresse IP à laquelle se lier. Pour les serveurs de catalogue et les serveurs conteneurs, définissez l'hôte et le port d'écoute dans le fichier des propriétés des serveurs. Ne pas spécifier l'adresse IP à utiliser produit des symptômes comme des dépassements du délai d'attente des connexions, des défaillances inhabituelles d'API et ce qui ressemble à des blocages des clients.

listenerPort
Indique le numéro de port auquel l'ORB (Object Request Broker) doit être associé. Cette propriété s'applique à la fois au serveur conteneur et au service de catalogue.

JMXServicePort
Indique le numéro de port sur lequel le serveur MBean doit écouter. Cette propriété s'applique à la fois au serveur conteneur et au service de catalogue.

Propriétés du serveur conteneur

statsSpec
Indique la spécification de statistiques du serveur conteneur.

Exemple :

```
all=disabled
```

memoryThresholdPercentage
Définit le seuil de mémoire pour l'expulsion basée sur la mémoire. Le pourcentage indique le segment de mémoire maximal qui doit être utilisé dans la machine virtuelle Java (JVM) avant expulsion. La valeur par défaut est `-1`, qui indique que le seuil de mémoire n'est pas défini. Si la propriété memoryThresholdPercentage est définie, la valeur MemoryPoolMXBean est définie à l'aide de la valeur fournie. Pour plus d'informations, voir l'interface MemoryPoolMXBean dans la spécification d'API Java. Toutefois, l'expulsion n'a lieu que si elle est activée sur un expulseur. Pour activer l'expulsion en fonction de la mémoire, voir les informations sur les expulseurs dans la rubrique *Présentation du produit*. Cette propriété ne s'applique qu'à un serveur conteneur.

catalogServiceEndPoints
Spécifie les points de contact à connecter au domaine de services de catalogue. Cette valeur doit être au format `hôte:port<,hôte:port>` où la valeur de l'hôte correspond à la valeur listenerHost et la valeur du port, à la valeur listenerPort du serveur de catalogues. Cette propriété ne s'applique qu'à un serveur conteneur.

Propriétés du service de catalogue

domainName
Spécifie le nom de domaine utilisé pour identifier de manière unique ce domaine de services de catalogue lors d'un routage vers plusieurs domaines. Cette propriété ne s'applique qu'au service de catalogue.

enableQuorum
Active le quorum pour le service de catalogue. Le quorum sert à garantir qu'une majorité du domaine de services de catalogue est disponible avant d'autoriser une modification du positionnement des partitions sur les serveurs conteneurs disponibles. Pour activer le quorum, spécifiez la valeur `true` ou `enabled`. La valeur par défaut est `disabled`. Cette propriété ne s'applique qu'au service de catalogue.

catalogClusterEndpoints
Spécifie les points de contact du domaine de services de catalogue pour le service de catalogue. Cette propriété spécifie les points de contact du service de catalogue pour démarrer le domaine de services de catalogue. Utilisez le format suivant :

```
serverName:hostName:clientPort:peerPort<serverName:hostName:clientPort:peerPort>
```

Cette propriété ne s'applique qu'au service de catalogue.

heartBeatFrequencyLevel
Indique la fréquence des signaux de présence. Le niveau de la fréquence des signaux de présence est un compromis entre l'utilisation des ressources

et la durée de reconnaissance des incidents. Plus les signaux de présence sont fréquents, plus une quantité importante de ressources est utilisée, mais les incidents sont détectés plus rapidement. Cette propriété ne s'applique qu'au service de catalogue. Utilisez l'une des valeurs suivantes :

- `0` : Indique un niveau de signal de présence à une fréquence standard. Avec cette valeur, la détection des basculements s'effectue à une fréquence raisonnable sans surutilisation des ressources. (Valeur par défaut)
- `-1` : Indique un niveau de signal de présence aggressif. Avec cette valeur, les incidents sont détectés plus rapidement, mais une quantité supérieure de ressources de processeur et de réseau est utilisée. Ce niveau est plus sensible aux signaux de présence manquants lorsque le serveur est occupé.
- `1` : Indique un niveau de signal de présence souple. Avec cette valeur, une fréquence réduite des signaux de présence augmente le délai de détection des incidents, mais réduit également l'utilisation du processeur et du réseau.

Propriétés du serveur de sécurité

Le fichier de propriétés du serveur est également utilisé pour configurer la sécurité du serveur eXtreme Scale. Vous utilisez un même fichier de propriétés de serveur pour spécifier les propriétés de base et les propriétés de sécurité.

Propriétés de sécurité générales

securityEnabled
Active la sécurité du serveur conteneur si la valeur est `true`. La valeur par défaut est `false`. Cette propriété doit correspondre à la propriété securityEnabled spécifiée dans le fichier `objectGridSecurity.xml` fourni au serveur de catalogues.

credentialAuthentication
Indique si ce serveur prend en charge l'authentification des données d'identification. Choisissez l'une des valeurs suivantes :

- `Jamais` : Le serveur ne prend pas en charge l'authentification des données d'identification.
- `Pris en charge` : Le serveur prend en charge l'authentification des données d'identification si le client le prend également.
- `Obligatoire` : Le client requiert l'authentification des données d'identification.

Pour plus de détails sur l'authentification des données d'identification, voir «Authentification du client d'application», à la page 360.

Paramètres de sécurité de la couche de transport

transportType
Indique le type de transport du serveur. Utilisez l'une des valeurs suivantes :

- `TCP/IP` : Indique que le serveur ne prend en charge que les connexions TCP/IP.
- `SSL pris en charge` : Indique que le serveur prend en charge les connexions TCP/IP et SSL (Secure Sockets Layer). (Valeur par défaut)
- `SSL requis` : Indique que le serveur requiert des connexions SSL.

Propriétés de configuration SSL

alias Indique le nom de l'alias dans le fichier de clés. Cette propriété est utilisée si le fichier de clés contient plusieurs certificats de paire de clés et que vous souhaitez sélectionner l'un des certificats.

Valeur par défaut : aucune

contextProvider
Indique le nom du fournisseur de contexte du service sécurisé. Si vous indiquez une valeur non valide, une exception de sécurité se produit et indique que le type du fournisseur de contexte est incorrect.

Les valeurs valides sont : `IBMJSSE2`, `IBMJSSE`, `IBMJSSEFIPS`, etc.

protocol
Indique le type du protocole de sécurité à utiliser pour le client. Définissez cette valeur de protocole en fonction du fournisseur JSSE (Java Secure Socket Extension) que vous utilisez. Si vous indiquez une valeur non valide, une exception de sécurité se produit et indique que la valeur du protocole est incorrecte.

Les valeurs valides sont : `SSL`, `SSLv2`, `SSLv3`, `TLS`, `TLSv1`, etc.

keyStoreType
Indique le type de fichier de clés. Si vous indiquez une valeur non valide, une exception de sécurité de l'environnement d'exécution se produit.

Les valeurs valides sont : `JKS`, `JCEK`, `PKCS12`, etc.

trustStoreType
Indique le type de fichier de clés certifiées. Si vous indiquez une valeur non valide, une exception de sécurité de l'environnement d'exécution se produit.

Les valeurs valides sont : `JKS`, `JCEK`, `PKCS12`, etc.

keyStore
Indique un chemin complet vers le fichier de clés.

Exemple :

```
etc/test/security/client.private
```

trustStore
Indique un chemin complet vers le fichier de clés certifiées.

Exemple :

```
etc/test/security/server.public
```

keyStorePassword
Indique le mot de passe de chaîne du fichier de clés. Vous pouvez coder cette valeur ou utiliser la valeur réelle.

trustStorePassword
Indique le mot de passe de chaîne du fichier de clés sécurisé. Vous pouvez coder cette valeur ou utiliser la valeur réelle.

clientAuthentication
Si cette propriété a la valeur true, le client SSL doit être authentifié. L'authentification du client SSL est différente de l'authentification du certificat du client. L'authentification du certificat client revient à authentifier un client dans un registre d'utilisateurs en fonction de la chaîne de certificats. Cette propriété garantit que le serveur se connecte au client approprié.

Paramètre SecureTokenManager

Le paramètre SecureTokenManager permet de protéger la chaîne secrète lors des authentification mutuelles de serveurs et pour protéger le jeton de connexion unique.«Sécurité de la grille», à la page 358

secureTokenManagerType
Indique le type de paramètre SecureTokenManager. Vous pouvez utiliser les paramètres suivants :

- `none` : Indique qu'aucun gestionnaire de jetons sécurisé n'est utilisé.
- `default` : Indique que le gestionnaire de jetons fourni avec le produit WebSphere eXtreme Scale est utilisé. Vous devez fournir une configuration de fichier de clés SecureToken.
- `custom` : Indique que vous possédez votre propre gestionnaire de jetons que vous avez spécifié avec la classe d'implémentation SecureTokenManager.

customTokenManagerClass
Indique le nom de votre classe d'implémentation SecureTokenManager, si vous avez spécifié la valeur `custom` pour la propriété SecureTokenManagerType. La classe d'implémentation doit avoir un constructeur par défaut à instancier.

customSecureTokenManagerProps
Indique les propriétés personnalisées de la classe d'implémentation SecureTokenManager. Cette propriété n'est utilisée que si la valeur secureTokenManagerType est `custom`. La valeur correspond à l'objet SecureTokenManager avec la méthode setProperties(String).

Configuration du fichier de clés à jeton sécurisé

secureTokenKeyStore
Indique le nom du chemin d'accès du fichier de clés qui stocke la paire de clés plublique-privée et la clé secrète.

secureTokenKeyStoreType
Indique le type de fichier de clés (par exemple, JCKES). Vous pouvez définir cette valeur en fonction du fournisseur JSSE (Java Secure Socket Extension) que vous utilisez. Toutefois, ce fichier de clés doit prendre en charge les clés secrètes.

secureTokenKeyPairAlias
Indique l'alias de la paire de clés publique-privée utilisée pour la signature et la vérification.

secureTokenKeyPairPassword
Indique le mot de passe permettant de protéger l'alias de paire de clés utilisé pour la signature et la vérification.

secureTokenSecretKeyAlias
Indique l'alias de clé secrète utilisé pour le chiffrement.

secureTokenSecretKeyPassword
Indique le mot de passe permettant de protéger la clé secrète.

secureTokenCipherAlgorithm
Indique l'algorithme utilisé pour fournir un chiffrement. Vous pouvez définir cette valeur en fonction du fournisseur JSSE (Java Secure Socket Extension) que vous utilisez.

secureTokenSignAlgorithm
Indique l'algorithme utilisé pour signer l'objet. Vous pouvez définir cette valeur en fonction du fournisseur JSSE que vous utilisez.

Chaîne d'authentification

authenticationSecret
Indique la chaîne secrète d'authentification du serveur. Lorsqu'un serveur démarre, il doit présenter cette chaîne au serveur président ou au serveur de catalogues. Si la chaîne secrète correspond à celle qui se touve sur le serveur président, ce serveur peut se connecter.

Configuration des ports

WebSphere eXtreme Scale est un cache réparti qui nécessite l'ouverture de ports pour communiquer avec l'ORB (Object Request Broker) et la pile TCP (Transmission Control Protocol) sur des machines virtuelles Java et d'autres machines.

Planification des ports réseau

WebSphere eXtreme Scale est un cache réparti devant ouvrir des ports pour communiquer avec l'ORB (Object Request Broker) et la pile TCP (Transmission Control Protocol) sur les machines virtuelles Java et les autres machines. Vous devez planifier et contrôler les ports, particulièrement dans un environnement de pare-feu, par exemple lorsque vous utilisez un service de catalogue et des conteneurs sur plusieurs ports.

Domaine de services de catalogue

Un domaine de services de catalogue nécessite que soient définis les ports suivants :

peerPort
Spécifie le port qui permet au gestionnaire de haute disponibilité (HA) de communiquer entre serveurs de catalogue homologues dans une pile TCP.

clientPort
Spécifie le port qui permet aux serveurs de catalogue d'accéder aux données des services de catalogue.

JMXServicePort
Spécifie le port que doit utiliser le service JMX (Java Management Extensions).

listenerPort
Définit le port d'écoute ORB qui permet aux conteneurs et aux clients de communiquer avec le service de catalogue via l'ORB.

La manière dont vous définissez ces ports dépend du mode que vous utilisez : autonome ou démarrage des serveurs de catalogue eXtreme Scale dans un environnement WebSphere Application Server :

- **En mode autonome :**

  Utilisez la commande startOgServer pour définir les ports indiqués plus haut avec l'option autonome, comme dans l'exemple suivant :

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

Voir «Démarrage du service de catalogue dans un environnement autonome», à la page 331 pour en savoir plus sur le démarrage du service de catalogue en mode autonome.

- **Pour un environnement WebSphere Application Server :**

  Vous pouvez définir un domaine de services de catalogue dans la console d'administration. Pour plus d'informations, voir «Création de domaines de service de catalogue dans WebSphere Application Server», à la page 346.

Serveurs conteneurs

Les serveurs conteneurs WebSphere eXtreme Scale requièrent également que plusieurs ports soient en fonctionnement. Par défaut, le serveur conteneur eXtreme Scale génère automatiquement son port de gestionnaire HA et son port d'écoute ORB avec des ports dynamiques. Dans un environnement de pare-feu, il est avantageux pour vous de planifier et de contrôler les ports. Certaines options vous permettent donc de démarrer les serveurs conteneurs eXtreme Scale avec le port du gestionnaire HA et le port d'écoute ORB à l'aide d'une option de la commande startOgServer, comme dans l'exemple ci-dessous :

```
-HaManagerPort <port_homologue>
-listenerPort <portORB>
```

Une planification optimale des ports est un avantage, mais il est difficile de les planifier et de les gérer lorsque des centaines de machines virtuelles Java sont démarrées dans une machine. Un conflit de port entraîne un échec de démarrage du serveur.

Lorsque la sécurité est activée, un port SSL (Secure Socket Layer) doit être ajouté aux ports répertoriés précédemment. L'utilisation de `Dcom.ibm.CSI.SSLPort=<sslPort>` comme argument **-jvmArgs** donne au port SSL la valeur de `<sslPort>`. Pour obtenir de l'aide sur la planification des ports, consultez les paramètres de sécurité dans eXtreme Scale.

Configuration de ports en mode autonome

Une machine virtuelle Java qui héberge une instance de service de catalogue requiert quatre ports. Deux ports sont utilisés en interne, le troisième est utilisé pour que les clients et les fragments de conteneur communiquent avec le protocole IIOP et le quatrième, pour les communications JMX (Java Management Extensions).

Pourquoi et quand exécuter cette tâche

Noeuds finaux de la machine virtuelle Java du service de catalogue

eXtreme Scale utilise le protocole IIOP principalement pour les communications entre les machines virtuelles Java. Les machines virtuelles Java du service de catalogue sont les seules machines virtuelles Java qui requièrent la configuration explicite des ports pour les services IIOP et qui regroupent les ports des services. Les ports internes sont spécifiés à l'aide de l'option de ligne de commande **-catalogServiceEndPoints** :

```
-catalogServiceEndPoints
<server:hôte:port:port,serveur:hôte:port:port>
```

Avec l'option de ligne de commande **-catalogServiceEndPoints**, vous pouvez configurer deux ports par serveur. Les ports IIOP sont configurés à l'aide des options de ligne de commande suivantes :

```
-listenerHost
<nom_hôte>
-listenerPort <port>
```

Lorsque chaque machine virtuelle Java de service de catalogue est démarrée, spécifiez l'ensemble complet des points de contact des services de cataloguer avec un port d'écoute unique pour cette machine virtuelle Java.

Noeuds finaux de la machine virtuelle Java du conteneur

Le conteneur machines virtuelles Java utilise deux ports. Un pour une utilisation interne et l'autre pour les communications IIOP. Généralement, les machines virtuelles Java du conteneur recherchent automatiquement les ports inutilisés, puis se configurent pour utiliser deux ports créés de manière dynamique. Cette procédure automatique minimise la configuration. Toutefois, si des pare-feux sont utilisés ou que vous souhaitez configurer les ports de manière explicite, vous pouvez utiliser l'option de ligne de commande pour spécifier le port d'ORB (Object Request Broker) à utiliser :

```
-listenerHost <nom_hôte>
-listenerPort <port>
```

Avec ces options de ligne de commande, vous pouvez indiquer le nom d'hôte (important pour l'association à la carte réseau appropriée) et le port à spécifier pour cette machine virtuelle Java. Vous pouvez spécifier le port interne à l'aide de l'argument de ligne de commande **-haManagerPort**. Toutefois, pour la configuration la plus simple, vous pouvez laisser l'environnement d'exécution choisir les ports.

Procédure

1. Démarrez le premier serveur de catalogues sur hostA. Exemple de commande :

   ```
   ./startOgServer.sh cs1 -listenerHost hostA -listenerPort 2809
   -catalogServiceEndPoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
   ```

2. Démarrez le second serveur de catalogues sur hostB. Exemple de commande :

   ```
   ./startOgServer.sh cs2 -listenerHost hostB -listenerPort 2809
   -catalogServiceEndPoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
   ```

   Les clients ont uniquement besoin de connaître les points de contact des programmes d'écoute de service de catalogue. Les clients extraient les points de contact des machines virtuelles Java des conteneurs, qui sont les machines virtuelles Java qui conservent les données, automatiquement à partir du service de catalogue. Pour se connecter au service de catalogue de l'exemple précédent, le client doit transmettre la liste suivante de paires `hôte:port` à l'API de connexion :

   ```
   hostA:2809,hostB:2809
   ```

   Pour démarrer la machine virtuelle Java d'un conteneur afin d'utiliser l'exemple de service de catalogue, utilisez la commande suivante :

   ```
   ./startOgServer.sh c0 -catalogServiceEndPoints hostA:2809,hostB:2809
   ```

Configuration de ports dans un environnement WebSphere Application Server

Le service de catalogue utilise une seule instance dans le gestionnaire de déploiement par défaut et utilise le port d'amorce du protocole IIOP (Internet Inter-ORB Protocol) de la machine virtuelle Java du gestionnaire de déploiement.

Pourquoi et quand exécuter cette tâche

Les applications Web ou EJB (Enterprise JavaBeans) d'une cellule peuvent se connecter à des grilles de la même cellule à l'aide de l'appel de connexion `null,null` au lieu de spécifier les ports d'amorçage du service de catalogue.

Si le domaine de services de catalogue dans WebSphere Application Server est hébergée par le gestionnaire de déploiement, les clients extérieurs à la cellule (y compris les clients Java Platform, Standard Edition) doivent se connecter au service de catalogue à l'aide du nom d'hôte et du port d'amorçage IIOP du gestionnaire de déploiement. Lorsque le service de catalogue s'exécute dans des cellules WebSphere Application Server alors que les clients sont exécutés hors des cellules, vous devez rechercher dans les pages de configuration des domaines eXtreme Scale, sur la console d'administration de WebSphere Application Server, les informations permettant de pointer un client sur le service de catalogue. Si vos clients se trouvent dans des cellules WebSphere Application Server, vous pouvez extraire les ports directement de l'interface CatalogServerProperties.

7.1+ Bien qu'il soit toujours possible d'utiliser la propriété `catalog.services.cluster` pour repérer les ports de connexion des clients, cette technique est à présent dépréciée. Si vous utilisez cette technique sans parvenir à trouver l'entrée `catalog.services.cluster`, utilisez le port IIOP dans le gestionnaire de déploiement pour la connexion des clients.

eXtreme Scale réutilise les ports DCS (Distribution and Consistency Services) du gestionnaire de haute disponibilité pour l'appartenance au groupe. Les ports JMX (Java Management Extensions) sont également réutilisés.

Configuration d'ORB

Le fichier orb.properties permet de transmettre les propriétés utilisées par l'ORB (Object Request Broker) afin de modifier le comportement du transport de la grille. WebSphere eXtreme Scale utilise l'ORB (Object Request Broker) pour activer la communication entre les processus. Aucune action n'est requise pour utiliser pour vos serveurs WebSphere eXtreme Scale l'ORB fourni par WebSphere eXtreme Scale ou par WebSphere Application Server. Nous allons exposer quelques points à prendre en considération pour l'optimisation des ORB ainsi que quelques tâches en rapport avec ces derniers.

Fichier de propriétés de l'ORB

Le fichier `orb.properties` est utilisé pour transmettre les propriétés utilisées par l'ORB (Object Request Broker) afin de modifier le comportement du transport de la grille.

Emplacement

Le fichier `orb.properties` se trouve dans le répertoire `java/jre/lib`. Si vous modifiez le fichier dans un répertoire `java/jre/lib` de WebSphere Application Server, les serveurs d'applications configurés sous cette installation utilisent également les paramètres du fichier.

Paramètres de référence

Les paramètres ci-après peuvent servir de référence, mais il ne s'agit pas nécessairement des meilleurs paramètres pour chaque environnement. Vous devez posséder une bonne connaissance de ces paramètres afin de bien choisir les valeurs appropriées dans votre environnement.

```
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
```

Descriptions des propriétés

Paramètres d'expiration

Les paramètres ci-après concernent le délai d'attente respecté par l'ORB avant d'abandonner des opérations de demande.

Délai d'expiration de la demande

Nom de la propriété : com.ibm.CORBA.RequestTimeout

Valeur : Entier pour le nombre de secondes.

Description : Indique pendant combien de secondes une demande doit attendre une réponse avant d'abandonner. Cette propriété influence la durée de la reprise en ligne du client en cas d'indisponibilité du réseau. Si vous spécifiez une valeur trop faible pour cette propriété, les demandes risquent d'arriver à expiration par inadvertance. Etudiez la valeur de cette propriété avec précaution pour empêcher ce cas de figure.

Délai d'expiration de la connexion

Nom de la propriété : com.ibm.CORBA.ConnectTimeout

Valeur : Entier pour le nombre de secondes.

Description : Indique pendant combien de secondes une tentative de connexion socket doit attendre avant d'abandonner. Cette propriété, comme celle du délai d'expiration de la demande, peut influencer la durée de la reprise en ligne du client en cas d'indisponibilité du réseau. En général, spécifiez une valeur inférieure à celle du délai d'expiration de la demande car le délai d'établissement d'une connexion doit être relativement constant.

Délai d'expiration des fragments

Nom de la propriété : com.ibm.CORBA.FragmentTimeout

Valeur : Entier pour le nombre de secondes.

Description : Indique pendant combien de secondes une demande de fragment doit attendre avant d'abandonner. Cette propriété est similaire à celle du délai d'expiration de la demande.

Paramètres du pool d'unités d'exécution

Ces propriétés restreignent la taille du pool d'unités d'exécution à un nombre spécifique d'unités d'exécution. Les unités d'exécution sont utilisées par l'ORB pour distribuer les demandes du serveur une fois qu'elles ont été reçues sur le socket. Si vous spécifiez des valeurs trop faibles pour ces propriétés, la longueur de la file d'attente des sockets augmente et les délais d'expiration risquent d'être dépassés.

Multiplicité des connexions

Nom de la propriété : com.ibm.CORBA.ConnectionMultiplicity

Valeur : Entier pour le nombre de connexions entre le client et le serveur. La valeur par défaut est 1. Si vous spécifiez une valeur supérieure, le multiplexage est défini entre plusieurs connexions.**Description :** Permet à l'ORB d'utiliser plusieurs connexions à un serveur. En théorie, la définition de cette valeur doit promouvoir le parallélisme par rapport aux connexions. En pratique, la multiplicité des connexions n'améliore pas les performances. Ne spécifiez pas ce paramètre.

Connexions ouvertes

Noms des propriétés : com.ibm.CORBA.MinOpenConnections, com.ibm.CORBA.MaxOpenConnections

Valeur : Entier pour le nombre de connexions.**Description :** Indique les nombres minimal et maximal de connexions ouvertes. L'ORB conserve un cache des connexions établies avec les clients. Ces connexions sont purgées lorsque la valeur com.ibm.CORBA.MaxOpenConnections est transmise. La purge des connexions peut nuire au comportement de la grille.

Peut augmenter

Nom de la propriété : com.ibm.CORBA.ThreadPool.IsGrowable

Valeur : Booléen acceptant la valeur `true` ou `false`.**Description :** Si cette propriété est activée, la taille du pool d'unités d'exécution utilisé par l'ORB pour les demandes entrantes peut être supérieure à la taille prise en charge par le pool. Si la taille du pool est dépassée, des unités d'exécution sont créées pour traiter la demande, mais elles ne sont pas placées dans un pool.

Longueur de la file d'attente des sockets de serveur

Nom de la propriété : com.ibm.CORBA.ServerSocketQueueDepth

Valeur : Entier pour le nombre de connexions.**Description :** Indique la longueur de la file d'attente des connexions entrantes des clients. L'ORB place les connexions entrantes des clients en file d'attente. Si la file d'attente est saturée, les connexions sont refusées. Le refus des connexions peut nuire au comportement de la grille.

Taille du fragment

Nom de la propriété : com.ibm.CORBA.FragmentSize

Valeur : Entier spécifiant le nombre d'octets. La valeur par défaut est 1024.**Description :** Indique la taille de paquet maximale utilisée par l'ORB lors de l'envoi d'une demande. Si la taille d'une demande est supérieure à la taille limite du fragment, la demande est divisée en fragments de demande qui sont envoyés séparément, puis réassemblés sur le serveur. La fragmentation des demandes est utile sur les réseaux non fiables où les packets doivent parfois être renvoyés. Toutefois, si le réseau est fiable, la division des demandes en fragments peut augmenter la charge du système.

Copies non locales

Nom de la propriété : com.ibm.CORBA.iiop.NoLocalCopies

Valeur : Booléen acceptant la valeur `true` ou `false`. **Description :** Indique si l'ORB utilise la transmission par référence. Par défaut, l'ORB utilise un appel de transmission par valeur. Un appel de transmission par valeur occupe davantage de place et augmente les coûts de sérialisation du chemin lorsqu'une interface est appelée en local. Si vous spécifiez la valeur true, l'ORB utilise une méthode de transmission par référence, qui est plus efficace que l'appel de transmission par valeur.

Intercepteurs non locaux

Nom de la propriété : com.ibm.CORBA.NoLocalInterceptors

Valeur : Booléen acceptant la valeur `true` ou `false`. **Description :** Indique si l'ORB appelle les intercepteurs de demande même pour les demandes locales (processus internes). Les intercepteurs utilisés par WebSphere eXtreme Scale pour la gestion de la sécurité et des routes ne sont pas obligatoires si la demande est gérée au sein du processus. Les intercepteurs qui circulent entre les processus ne sont requis que pour les opérations RPC (Remote Procedure Call). En définissant des intercepteurs non locaux, vous pouvez éviter la charge supplémentaire due à l'utilisation d'intercepteurs locaux.

Avertissement : Si la sécurité de WebSphere eXtreme Scale est activée, donnez la valeur `false` à la propriété com.ibm.CORBA.NoLocalInterceptors. L'infrastructure de sécurité utilise des intercepteurs pour l'authentification.

Si vous ne souhaitez pas appliquer la sécurité du transport entre les clients et les serveurs ObjectGrid, vous devez ajouter davantage de propriétés au fichier `orb.properties`. Pour plus d'informations sur ces propriétés, reportez-vous à la section sur le fichier `orb.properties` pour la prise en charge de la sécurité du transport, dans la rubrique «Protocole TLS et couche de connexion sécurisée», à la page 365.

Propriétés ORB et paramétrage du descripteur de fichiers

L'optimisation inclut également les propriétés Object Request Broker (ORB) et le paramétrage du descripteur de fichiers.

Propriétés ORB

L'ORB sert à WebSphere eXtreme Scale pour communiquer dans l'ensemble d'une pile TCP. L'indispensable fichier `orb.properties` se trouve dans le répertoire `java/jre/lib`. Dans le cas de lourde charge d'objets de grande taille, activez la fragmentation ORB en spécifiant le paramètre suivant :

```
com.ibm.CORBA.FragmentSize=<taille appropriée>
```

Empêchez la croissance du pool d'unités d'exécution en spécifiant le paramètre suivant :

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

En spécifiant le paramètre suivant, fixez des délais d'expiration appropriés qui vous éviteront d'avoir un trop grand nombre d'unités d'exécution dans des situations anormales :

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

- com.ibm.CORBA.FragmentTimeout

Descripteur de fichiers

Les systèmes UNIX® et Linux imposent une limite au nombre de fichiers ouverts autorisé par processus. C'est le système d'exploitation qui spécifie le nombre permis de fichiers ouverts. Si cette valeur est trop basse, une erreur d'allocation de mémoire se produira sur AIX avec consignation d'une erreur de fichiers ouverts en trop grand nombre.

Dans la fenêtre de terminal UNIX, augmentez cette valeur au-dessus de la valeur par défaut du système. Dans le cas de grosses machines SMP avec des clones, fixez une valeur illimitée.

Pour les configurations AIX, fixez une valeur de `-1` (illimitée) à l'aide de la commande `ulimit -n -1`.

Pour les configurations Solaris, fixez une valeur de `16384` (illimitée) à l'aide de la commande `ulimit -n 16384`.

Pour connaître la valeur actuelle, utilisez la commande `ulimit –a`.

Activation de NIO ou de ChannelFramework sur l'ORB

WebSphere eXtreme Scale fournit une propriété de serveur permettant de définir le mode de transport comme étant ChannelFramework afin d'activer NIO (les entrées-sorties non bloquantes) sur l'ORB.

Avant de commencer

Trouvez le fichier de propriétés du serveur ou créez-en un. Vous pouvez activer ChannelFramework sur le service de catalogue et sur les serveurs conteneurs. pour plus de détails, voir Fichier de propriétés de serveur.

Pourquoi et quand exécuter cette tâche

A l'heure actuelle, il est possible de fonctionner avec NIO ou ChannelFramework) dans des scénarios WebSphere eXtreme Scale autonome. Pour pouvoir fonctionner avec NIO dans l'ORB d'IBM, vous devez définir ChannelFramework comme mode de transport de cet ORB. En effet, par défaut cet ORB fonctionne en mode Pluggable. WebSphere eXtreme Scale fournit une propriété de serveur permettant de définir le mode de transport comme ChannelFramework.

Important :

Les versions actuelles de WebSphere Application Server ne prennent en charge que le mode Pluggable. Lorsque WebSphere eXtreme Scale s'exécute intégré à WebSphere Application Server, il doit se conformer au mode Pluggable. WebSphere eXtreme Scale utilisant également le SSL/Transport Security de WebSphere Application Server, lui aussi ne prend en charge que le mode Pluggable.

Procédure

1. Ajoutez la propriété enableChannelFramework=true au fichier de propriétés de votre serveur.
2. Assurez-vous que ce fichier de propriétés ne contredit pas le fichier des propriétés de l'ORB.

Si le fichier de propriétés du serveur active le mode de transport ChannelFramework, mais que ce dernier est défini comme Pluggable dans le fichier orb.properties, le paramétrage du serveur ne prendra pas le pas sur celui de l'ORB. Un message vous avertira dans le journal qu'il existe deux paramétrages. Pour permettre à la propriété enableChannelFramework=true d'entrer en vigueur, ajustez les propriétés qui indiquent que le mode de transport est Pluggable : remplacez com.ibm.CORBA.TransportMode=Pluggable par com.ibm.CORBA.TransportMode=ChannelFramework ou supprimez carrément la propriété.

3. Fournissez le fichier de propriétés du serveur modifié au démarrage du service de catalogue ou du serveur conteneur. Pour plus de détails sur l'utilisation des fichiers de propriétés de serveur pour démarrer un serveur, voir Fichier de propriétés de serveur.

Résultats

Lorsqu'un service de catalogue ou un serveur conteneur utilise le mode de transport channelFramework, il écrit le message suivant dans le journal.

```
CWOBJ0052I: La propriété IBM ORB TransportMode a été associée à ChannelFramework.
```

Si vous voyez dans le journal le message qui suit, examinez les propriétés de votre ORB, comme expliqué précédemment.

```
CWOBJ0055W: La propriété IBM ORB TransportMode a été associée à ChannelFramework
dans le fichiers des propriétés du serveur mais le fichier orb.properties
existant contient déjà un paramètre TransportMode défini. La propriété
TransportMode ne sera pas remplacée.
```

Notez que lorsque vous activez ChannelFramework, la valeur maximum de ServerSocketQueueDepth est de 512. Si le paramètre ServerSocketQueueDepth d'orb.properties est supérieur à 512, le serveur ramènera automatiquement ServerSocketQueueDepth à 512 et vous en avisera en écrivant un message d'information dans le journal. Aucune action de votre part n'est nécessaire.

```
CWOBJ0053I: La propriété IBM ORB ServerSocketQueueDepth a été associée
à 512 pour s'exécuter correctement avec la propriété
ChannelFramework TransportMode.
```

Utilisation d'Object Request Broker avec des processus WebSphere eXtreme Scale autonomes

Vous pouvez utiliser WebSphere eXtreme Scale avec des applications qui utilisent Object Request Broker (ORB) directement dans des environnements ne contenant pas WebSphere Application Server ou WebSphere Application Server Network Deployment.

Avant de commencer

Si vous utilisez l'ORB dans le même processus que eXtreme Scale lorsque vous exécutez des applications, ou d'autres composants et infrastructures, non inclus avec eXtreme Scale, il se peut que vous deviez effectuer des tâches supplémentaires pour vous assurer que eXtreme Scale fonctionne correctement dans votre environnement.

Pourquoi et quand exécuter cette tâche

Ajoutez la propriété **ObjectGridInitializer** au fichier `orb.properties` pour initialiser l'utilisation de l'ORB dans votre environnement. Utilisez l'ORB pour

activer la communication entre les processus eXtreme Scale et les autres processus de votre environnement. Le fichier `orb.properties` se trouve dans le répertoire `java/jre/lib`. Pour les descriptions des propriétés et des paramètres, voir «Fichier de propriétés de l'ORB», à la page 195.

Procédure

Entrez la ligne suivante et sauvegardez vos modifications :

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

Résultats

eXtreme Scale initialise correctement l'ORB et coexiste avec d'autres applications pour lesquelles l'ORB est activé.

Pour utiliser une version personnalisée de l'ORB avec eXtreme Scale, voir «Configuration d'un ORB personnalisé».

Configuration d'un ORB personnalisé

WebSphere eXtreme Scale utilise l'ORB (Object Request Broker) pour activer les communications entre les processus. Aucune action n'est requise pour utiliser pour vos serveurs WebSphere eXtreme Scale l'ORB fourni par WebSphere eXtreme Scale ou par WebSphere Application Server. L'utilisation des mêmes ORB pour vos clients WebSphere eXtreme Scale ne vous demandera guère plus. En revanche, si vous devez utiliser un ORB personnalisé, celui qui est fourni avec le IBM SDK est un bon choix, bien qu'il nécessite un peu de configuration, que nous allons expliquer ici. Il est également possible d'utiliser d'autres ORB fournis par d'autres constructeurs, là aussi moyennant quelque configuration.

Avant de commencer

Vous devez commencer par décider quel ORB vous allez utiliser : celui fourni avec WebSphere eXtreme Scale ou avec WebSphere Application Server, celui fourni avec le IBM SDK ou un ORB tierce partie.

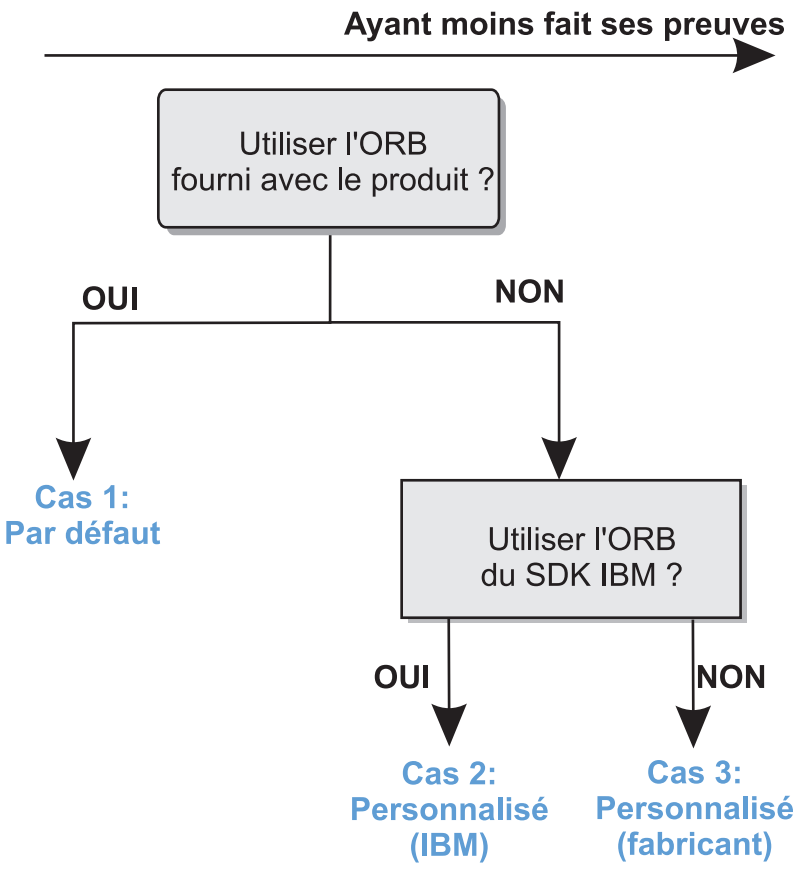


Figure 10. Choix de l'ORB

Vous n'êtes pas obligé de faire les mêmes choix pour les processus serveur WebSphere eXtreme Scale et les processus client WebSphere eXtreme Scale. eXtreme Scale prend en charge les kits de développeur de la plupart des fournisseurs, mais il est recommandé d'utiliser pour vos processus, tant serveur que client, l'ORB fourni avec eXtreme Scale. eXtreme Scale ne prend pas en charge l'ORB fourni avec le Java Development Kit (JDK) de Sun Microsystems.

Pourquoi et quand exécuter cette tâche

Avant d'utiliser l'ORB que vous avez choisi, familiarisez-vous avec la configuration requise.

Cas n° 1 : ORB par défaut

- Pour vos processus serveur WebSphere eXtreme Scale, aucune configuration n'est requise pour utiliser l'ORB fourni avec WebSphere eXtreme Scale ou avec WebSphere Application Server.
- Pour vos processus client WebSphere eXtreme Scale, un minimum de configuration du chemin d'accès aux classes est requis pour pouvoir utiliser l'ORB fourni avec WebSphere eXtreme Scale ou avec WebSphere Application Server.

Cas n° 2 : ORB personnalisé (IBM)

Pour configurer vos processus client WebSphere eXtreme Scale afin qu'ils utilisent l'ORB fourni avec le IBM SDK, voyez les instructions plus bas. Vous pouvez utiliser l'ORB IBM, que vous utilisiez le IBM SDK ou un autre kit de développement.

L'utilisation du IBM SDK version 5 (ou plus récente) requiert moins de configuration que celui de version 1.4.2.

Cas n° 3: ORB personnalisé (fabricant)
L'utilisation d'un ORB tierce partie pour les processus client WebSphere eXtreme Scale est l'option qui a fait le moins l'objet de tests chez IBM. Avant de contacter le support technique IBM, vous devez vous assurer que les problèmes rencontrés en utilisant des ORB d'éditeurs de logiciels indépendants sont bien reproductibles avec l'ORB IBM ORB et un JRE compatible.

L'ORB fourni avec le Java Development Kit (JDK) de Sun Microsystems n'est pas pris en charge.

Procédure

- Configurez vos processus client pour qu'ils utilisent l'un des ORB par défaut **(cas n° 1)**.

  `-jvmArgs –Djava.endorsed.dirs=`*`répertoire_ORB_défaut`*

- Configurez les processus client ou serveur pour qu'ils utilisent le IBM SDK version 5 **(cas n° 2)**.
  1. Copiez les fichiers JAR de l'ORB dans un répertoire vide (que nous désignerons sous le nom de *répertoire_ORB_personnalisé*.
     - `ibmorb.jar`
     - `ibmorbapi.jar`

     Conseil : Si vous utilisez un ORB personnalisé fourni par une tierce partie **(cas n° 3)**, ces fichiers JAR supplémentaires risquent d'être nécessaires :
     - `ibmext.jar`
     - `ibmcfw.jar` (pour l'ORB NIO)
  2. Dans les scripts qui lancent la commande Java, spécifiez le *répertoire_ORB_personnalisé* comme répertoire endorsed.

     Conseil : Si vos commandes Java se réfèrent déjà à un répertoire endorsed, vous avez la possibilité de faire du *répertoire_ORB_personnalisé* un sous-répertoire du répertoire endorsed existant, ce qui vous évitera de modifier les scripts. Si vous préférez quand même modifier ces derniers, n'oubliez pas de faire suivre l'argument `–Djava.endorsed.dirs=` existant de *custom_ORB_directory* plutôt que de le remplacer complètement.
     - Modifiez les scripts pour un environnement eXtreme Scale autonome.

       Dans le fichier `setupCmdLine.bat|sh`, modifiez le chemin pour la variable *OBJECTGRID_ENDORSED_DIRS* pour qu'il fasse référence au *répertoire_ORB_personnalisé*. Sauvegardez vos modifications.
     - Modifiez les scripts lorsque'eXtreme Scale est imbriqué dans un environnement WebSphere Application Server.

       Ajoutez la propriété système et les paramètres suivants au script `startOgServer` :

       `-jvmArgs –Djava.endorsed.dirs=`*`répertoire_ORB_personnalisé`*
     - Modifiez les scripts personnalisés qui vous servent à démarrer un processus d'application client ou un processus serveur.

       `-Djava.endorsed.dirs=`*`répertoire_ORB_personnalisé`*

- Configurez les processus client ou serveur pour qu'ils utilisent le IBM SDK version 1.4.2 **(cas n° 2)**. Si votre environnement contient un SDK Version 1.4.2, intégrez l'ORB IBM dans le SDK spécifié.

1. Téléchargez et extrayez l'ORB d'un IBM SDK version 1.4.2.

   Si aucun IBM SDK n'est disponible pour votre plateforme, téléchargez et extrayez le IBM Developer Kit de Linux, Java Technology Edition. Voir IBM developer kits.
2. Copiez les fichiers JAR de l'ORB vers le SDK cible. Copiez les fichiers `java/jre/lib/ibmorb.jar` et `java/jre/lib/ibmorbapi.jar` dans le répertoire `java/jre/lib/ext` sur le SDK cible.
3. Modifiez les propriétés de l'ORB. Créez ou éditez le fichier `orb.properties`, qui se trouve dans le répertoire `java/jre/lib` du SDK. Ajoutez les propriétés suivantes ou vérifiez que les propriétés suivantes existent dans le fichier :

   ```
   org.omg.CORBA.ORBClass=com.ibm.CORBA.iiop.ORB
   org.omg.CORBA.ORBSingletonClass=com.ibm.rmi.corba.ORBSingleton
   ```

   Pour les descriptions des propriétés et des paramètres, voir «Fichier de propriétés de l'ORB», à la page 195.
4. Assurez-vous que l'analyseur syntaxique XML est bien disponible.
   - Téléchargez Xerces2 Java 2.9 depuis la page The Apache Xerces Project - Downloads.
   - Repérez les fichiers `xercesImpl.jar` et `xml-apis.jar`.
   - Copiez ces fichiers vers le répertoire `lib/ext`.

Configuration des clients

Vous pouvez configurer WebSphere eXtreme Scale pour qu'il s'exécute dans un environnement autonome, tout comme vous pouvez le configurer pour qu'il s'exécute dans un environnement avec WebSphere Application Server or WebSphere Application Server Network Deployment. Pour qu'un déploiement eXtreme Scale extrait les changements de configuration côté grille du serveur, vous devez redémarrer les processus pour appliquer ces changements au lieu d'attendre qu'ils soient appliqués de manière dynamique. Toutefois, côté client, vous ne pouvez pas modifier les paramètres de configuration d'une instance de client existante, mais vous pouvez créer un client avec les paramètres dont vous avez besoin à l'aide d'un fichier XML ou d'un programme. Lorsque vous créez un client, vous pouvez remplacer les paramètres par défaut provenant de la configuration de serveur actuelle.

Vous pouvez configurer un client eXtreme Scale des différentes manières suivantes, dont chacune peut être effectuée par programmation ou à l'aide d'un fichier XML de substitution par le client :

- configuration XML
- configuration par programmation
- configuration Spring Framework
- désactivation du cache local

Vous pouvez remplacer les plug-in suivants sur un client :

- **plug-in ObjectGrid**
  - plug-in TransactionCallback
  - plug-in ObjectGridEventListener
- **plug-in BackingMap**
  - plug-in Evictor
  - plug-in MapEventListener
  - attribut numberOfBuckets

– attribut ttlEvictorType
– attribut timeToLive

Fichier de propriétés du client

Vous pouvez créer un fichier de propriétés en fonction de vos exigences pour les processus du client eXtreme Scale.

Exemples de fichier de propriétés du client

Vous pouvez utiliser le fichier `sampleClient.properties` qui se trouve dans le répertoire *racine_extremescale*`/properties` pour créer votre fichier de propriétés.

Spécification d'un fichier de propriétés du client

Vous pouvez spécifier le fichier de propriétés du client de l'une des manières ci-après. La spécification d'un paramètre en utilisant l'un des éléments plus loin dans la liste remplace le paramètre précédent. Par exemple, si vous spécifiez une valeur de propriété système pour le fichier de propriétés du client, les propriétés de ce fichier remplacent les valeurs du fichier `objectGridClient.properties` qui se trouvent dans le chemin d'accès aux classes.

1. Comme fichier nommé correctement dans le chemin d'accès aux classes. Vous ne pouvez pas placer ce fichier dans le répertoire actuel du système :

   ```
   objectGridClient.properties
   ```

2. Comme propriété système dans une configuration autonome ou WebSphere Application Server. Cette valeur peut spécifier un fichier du répertoire actuel du système, mais non un fichier du chemin d'accès aux classes :

   ```
   -Dobjectgrid.client.props=nom_fichier
   ```

3. Comme une substitution à l'aide d'un programme, à l'aide de la méthode ClientClusterContext.getClientProperties. Les données de l'objet sont alimentées avec celles des fichiers de propriétés. Vous ne pouvez pas configurer les propriétés de sécurité à l'aide de cette méthode.

Propriétés du client

7.1+ **listenerHost**
Indique le nom d'hôte auquel l'ORB (Object Request Broker) doit être associé.

Si votre configuration implique la présence de plusieurs cartes réseau, définissez l'hôte et le port d'écoute pour faire connaître à l'ORB de la machine virtuelle Java l'adresse IP à laquelle se lier. Pour le client, utilisez le fichier de propriétés de ce dernier. Ne pas spécifier l'adresse IP à utiliser produit des symptômes comme des dépassements du délai d'attente des connexions, des défaillances inhabituelles d'API et ce qui ressemble à des blocages des clients.

7.1+ **listenerPort**
Indique le numéro de port auquel l'ORB (Object Request Broker) doit être associé.

preferLocalProcess
Indique si le processus local est recommandé pour le routage. Si vous spécifiez la valeur true, les demandes sont routées vers des fragments placés dans le même processus que le client, si nécessaire.

Valeur par défaut : `true`

preferLocalHost
Indique si l'hôte local est recommandé pour le routage. Si vous spécifiez la valeur true, les demandes sont routées vers des fragments placés sur le même hôte que le client, si nécessaire.

Valeur par défaut : `true`

preferZones
Indique une liste de zones de routage recommandées. Chaque zone spécifiée est séparée par une virgule, au format suivant : `preferZones=ZoneA,ZoneB,ZoneC`

Valeur par défaut : aucune

requestRetryTimeout
Indique combien de temps une demande doit être renouvelée (en millisecondes). Utilisez l'une des valeurs admises suivantes :

- Une valeur égale à `0` indique que la demande doit échouer rapidement et ignorer la logique interne relative aux nouvelles tentatives.
- Une valeur égale à `-1` indique que le délai entre les tentatives de la demande n'est pas défini, ce qui signifie que la durée de la demande dépend du délai d'expiration des transactions. (Valeur par défaut)
- Une valeur supérieure à `0` indique la valeur du délai d'expiration de la demande en millisecondes. Les exceptions qui ne peuvent pas être résolues, même après une nouvelle tentative (par exemple, une exception DuplicateException), sont renvoyées immédiatement. Le délai de transaction est toujours utilisé comme délai d'attente maximal.

Propriétés de sécurité du client

Propriétés de sécurité générales

securityEnabled
Active la sécurité du client WebSphere eXtreme Scale. Ce paramètre de sécurité doit correspondre au paramètre securityEnabled du fichier de propriétés du serveur WebSphere eXtreme Scale. Si les paramètres ne correspondent pas, une exception est générée.

Valeur par défaut : `false`

Propriétés de configuration de l'authentification des données d'identification

credentialAuthentication
Indique la prise en charge de l'authentification des données d'identification du client. Utilisez l'une des valeurs admises suivantes :

- `Jamais` : Le client ne prend pas en charge l'authentification des données d'identification.
- `Pris en charge` : Le client prend en charge l'authentification des données d'identification si le serveur le prend également. (Valeur par défaut)
- `Obligatoire` : Le client requiert l'authentification des données d'identification.

authenticationRetryCount
Indique le nombre de tentatives d'authentification si les données d'identification sont arrivées à expiration. Si cette valeur est égale à 0, les tentatives d'authentification ne sont pas renouvelées.

Valeur par défaut : 3

credentialGeneratorClass
Indique le nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Cette classe est utilisée pour obtenir les données d'identification des clients.

Valeur par défaut : aucune

credentialGeneratorProps
Indique les propriétés de la classe d'implémentation CredentialGenerator. Les propriétés correspondent à l'objet avec la méthode setProperties(String). La valeur credentialGeneratorprops n'est utilisée que si la valeur de la propriété credentialGeneratorClass n'est pas null.

Propriétés de configuration de la sécurité de la couche de transport

transportType
Indique le type de transport du client. Les valeurs possibles sont :

- `TCP/IP` : Indique que le client ne prend en charge que les connexions TCP/IP.
- `SSL pris en charge` : Indique que le client prend en charge les connexions TCP/IP et SSL (Secure Sockets Layer). (Valeur par défaut)
- `SSL requis` : Indique que le client requiert des connexions SSL.

Propriétés de configuration SSL

alias Indique le nom de l'alias dans le fichier de clés. Cette propriété est utilisée si le fichier de clés contient plusieurs certificats de paire de clés et que vous souhaitez sélectionner l'un des certificats.

Valeur par défaut : aucune

contextProvider
Indique le nom du fournisseur de contexte du service sécurisé. Si vous indiquez une valeur non valide, une exception de sécurité se produit et indique que le type du fournisseur de contexte est incorrect.

Les valeurs valides sont : `IBMJSSE2`, `IBMJSSE`, `IBMJSSEFIPS`, etc.

protocol
Indique le type du protocole de sécurité à utiliser pour le client. Définissez cette valeur de protocole en fonction du fournisseur JSSE (Java Secure Socket Extension) que vous utilisez. Si vous indiquez une valeur non valide, une exception de sécurité se produit et indique que la valeur du protocole est incorrecte.

Les valeurs valides sont : `SSL`, `SSLv2`, `SSLv3`, `TLS`, `TLSv1`, etc.

keyStoreType
Indique le type de fichier de clés. Si vous indiquez une valeur non valide, une exception de sécurité de l'environnement d'exécution se produit.

Les valeurs valides sont : `JKS`, `JCEK`, `PKCS12`, etc.

trustStoreType
Indique le type de fichier de clés certifiées. Si vous indiquez une valeur non valide, une exception de sécurité de l'environnement d'exécution se produit.

Les valeurs valides sont : `JKS`, `JCEK`, `PKCS12`, etc.

keyStore
Indique un chemin complet vers le fichier de clés.

Exemple :

```
etc/test/security/client.private
```

trustStore
Indique un chemin complet vers le fichier de clés certifiées.

Exemple :

```
etc/test/security/server.public
```

keyStorePassword
Indique le mot de passe de chaîne du fichier de clés. Vous pouvez coder cette valeur ou utiliser la valeur réelle.

trustStorePassword
Indique le mot de passe de chaîne du fichier de clés sécurisé. Vous pouvez coder cette valeur ou utiliser la valeur réelle.

Configuration de clients avec WebSphere eXtreme Scale

Vous pouvez configurer un client eXtreme Scale en fonction de vos besoins, par exemple modifier certains paramètres.

Configuration du client avec XML

Vous pouvez utiliser un fichier XML ObjectGrid pour modifier les paramètres côté client. Pour modifier les paramètres d'un client eXtreme Scale , vous devez créer un fichier XML ObjectGrid ayant une structure semblable à celle du fichier utilisé pour le serveur eXtreme Scale.

Nous supposerons que le fichier XML suivant a été couplé à un fichier XML de règle de déploiement et que ces fichiers ont été utilisés pour démarrer un serveur 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>
```

Sur un serveur eXtreme Scale, l'instance ObjectGrid nommée CompanyGrid se comporte conformément au comportement défini par le fichier `companyGridServerSide.xml`. Par défaut, les paramètres du client CompanyGrid sont identiques à ceux de l'instance CompanyGrid qui s'exécute sur le serveur. Certains paramètres du client peuvent cependant être remplacés, comme suit :

1. Créez une instance ObjectGrid propre au client.
2. Copiez le fichier XML ObjectGrid utilisé pour ouvrir le serveur.
3. Editez le nouveau fichier pour personnaliser le côté client.
   - Pour définir ou mettre à jour les attributs du client, indiquez une nouvelle valeur ou modifiez la valeur existante.
   - Pour supprimer un plug-in du client, utilisez la chaîne vide comme valeur pour l'attribut className.
   - Pour modifier un plug-in existant, indiquez une nouvelle valeur pour l'attribut className.
   - Vous pouvez aussi ajouter les plug-in pris en charge pour les remplacements par les clients : TRANSACTION_CALLBACK, OBJECTGRID_EVENT_LISTENER, EVICTOR et MAP_EVENT_LISTENER.
4. Créez un client à l'aide du nouveau fichier XML de remplacement par les clients.

Le fichier XML ObjectGrid suivant peut être utilisé pour définir certains attributs et plug-in du client 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>
```

- Le TransactionCallback du client est com.company.MyClientTxCallback et non le paramètre com.company.MyTxCallback du côté serveur.
- Le client n'est associé à aucun plug-in ObjectGridEventListener car la valeur className est la chaîne vide.
- Le client associe numberOfBuckets à la valeur 1429 pour le backingMap Customer, conserve le plug-in Evictor et supprime le plug-in MapEventListener.

- Les attributs numberOfBuckets et timeToLive du backingMap OrderLine ont été modifiés.
- Bien qu'un attribut lockStrategy différent ait été indiqué, les conséquences sont nulles car cet attribut n'est pas pris en charge pour un remplacement par le client.

Pour créer le client CompanyGrid à l'aide du fichier `companyGridClientSide.xml`, transmettez le fichier XML ObjectGrid en tant qu'URL à l'une des méthodes de connexion de l'ObjectGridManager.

Création du client pour XML

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

Configuration du client à l'aide d'un programme

Vous pouvez aussi remplacer les paramètres ObjectGrid côté client à l'aide d'un programme. Créez un objet ObjectGridConfiguration dont la structure est semblable à celle de l'instance ObjectGrid côté serveur. Le code suivant crée une instance ObjectGrid côté client qui est fonctionnellement équivalente au remplacement par le client de la section précédente où un fichier XML était utilisé.

Remplacement côté client à l'aide d'un programme

```
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);
```

L'instance ogManager de l'interface ObjectGridManager ne vérifie les remplacements que dans les objets ObjectGridConfiguration et BackingMapConfiguration inclus dans la mappe overrideMap. Par exemple, le code précédent remplace le nombre de compartiments de la mappe OrderLine. La mappe Order reste cependant inchangée côté client car aucune configuration de cette mappe n'est incluse.

Configuration du client dans Spring

Les paramètres ObjectGrid côté client peuvent également être remplacés à l'aide de Spring Framework. L'exemple de fichier XML suivant montre comment générer un élément ObjectGridConfiguration et l'utiliser pour remplacer certains paramètres côté client. Cet exemple appelle les mêmes API que celles montrées dans la configuration par programmation. Cet exemple est également fonctionnellement équivalent à l'exemple de la configuration XML ObjectGrid.

Configuration du client avec 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>
```

Après avoir créé le fichier XML, chargez le fichier et générez l'ObjectGrid à l'aide du fragment de code suivant :

```
BeanFactory beanFactory = new XmlBeanFactory(new

UrlResource("file:test/companyGridSpring.xml"));

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

Pour plus d'informations sur la création d'un fichier XML de descripteur, lire Présentation générale de l'intégration à Spring.

Désactivation du cache local du client

Le cache local est activé par défaut lorsque le verrouillage est configuré sur OPTIMISTIC ou sur NONE. Lorsqu'il est configuré sur PESSIMISTIC, le cache n'est pas activé. Pour désactiver le cache local, vous devez donner la valeur 0 à l'attribut numberOfBuckets dans le fichier de descripteur des remplacements ObjectGrid par les clients.

Activation du mécanisme d'invalidation du client

Dans un environnement WebSphere eXtreme Scale réparti, le côté client dispose par défaut d'un cache local lorsqu'il utilise la stratégie de verrouillage optimiste ou lorsque le verrouillage est désactivé. Ce cache contient ses propres données locales. Si un client eXtreme Scale valide une mise à jour, celle-ci est envoyée au cache local du client et au serveur. Toutefois, les autres clients eXtreme Scale ne reçoivent pas les informations relatives à cette mise à jour et leurs données risquent de devenir obsolètes.

Cache local

Les applications doivent être informées de l'éventuelle présence de données obsolètes dans le client eXtreme Scale. Vous pouvez utiliser la classe ObjectGridEventListener JMS (Java Message Service) pré-intégrée, com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener, pour

activer le mécanisme d'invalidation du client dans un environnement eXtreme Scale connu en tant que grille eXtreme Scale.

Le mécanisme d'invalidation du client permet de résoudre les problèmes liés à la présence de données obsolètes dans le cache local du client en environnement eXtreme Scale réparti. Ce mécanisme vérifie que la mémoire est synchronisée avec les serveurs ou les autres clients. Toutefois, même lorsque ce mécanisme existe, la mise à jour de la mémoire n'est pas immédiate. Lorsque l'environnement d'exécution de eXtreme Scale publie des mises à jour, un délai est généré.

Il existe deux modèles pour le mécanisme d'invalidation du client dans un environnement eXtreme Scale réparti :

- Modèle client-serveur : tous les processus serveur ont un rôle de diffuseur de publications qui publie toutes les modifications transactionnelles vers la destination JMS désignée. Tous les processus client ont un rôle de récepteur : ils reçoivent toutes les modifications transactionnelles à partir de la destination JMS désignée.
- Modèle client ayant les deux rôles : aucune interaction n'existe entre les processus serveur et la destination JMS. Tous les processus client ont le rôle de diffuseur de publications JMS et de récepteur. Les modifications transactionnelles effectuées sur le client sont publiées vers la destination JMS et tous les clients les reçoivent.

Pour plus d'information, consultez «Programme d'écoute d'événement JMS», à la page 139.

Modèle client-serveur

Dans un modèle client-serveur, les serveurs ont le rôle de diffuseur de publications JMS et le client a le rôle de récepteur JMS.

Exemple XML de modèle client-serveur

```
<?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>
```

Modèle client ayant les deux rôles

Dans ce modèle, chaque client a le rôle de diffuseur de publications JMS et de récepteur. Le client publie chaque modification transactionnelle validée vers une destination JMS désignée et reçoit toutes les modifications des autres clients. Aucune interaction n'a lieu entre le serveur et JMS.

Exemple XML de modèle où le client a les deux rôles

```
<?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>
```

Configuration du délai entre deux nouvelles tentatives de demandes

Avec des mappes fiables, il est possible d'indiquer à WebSphere eXtreme Scale pendant combien de temps effectuer de nouvelles tentatives pour des demandes de transactions (retry timeout).

Les mappes fiables peuvent être configurées de deux manières. Si la valeur est supérieure à zéro, la demande est réessayée jusqu'à ce que le délai spécifié soit dépassé ou qu'une défaillance permanente (exception DuplicateKeyException, par exemple) soit rencontrée. Une valeur de zéro indique le mode fail-fast et eXtreme Scale n'effectue pas de nouvelles tentatives.

Le délai d'expiration est à indiquer en millisecondes dans le fichier des propriétés du client ou dans un objet Session. L'objet Session a la précédente sur les propriétés du client. Pendant l'exécution, le délai d'expiration des transactions est utilisé avec celui des nouvelles tentatives, ce qui garantit que ce dernier n'excède pas le premier.

En raison des variations dans la manière dont s'effectuent les transactions autocommit et non autocommit (celles qui utilisent les méthodes explicites begin et commit) les exceptions valides pour le retry sont différentes.

Dans le cas de transactions appelées au sein d'une session, le retry est valide pour les exceptions SystemExceptions de CORBA et TargetNotAvailable d'eXtreme Scale.

Dans le cas de transactions à validation automatique, le retry est valide pour les exceptions CORBA SystemExceptions eXtreme Scale Availability (ReplicationVotedToRollbackTransactionException, TargetNotAvailable, AvailabilityException, et autres).

Pour plus d'informations, voir dans le *Guide de programmation* la rubrique consacrée à l'utilisation d'objets Session pour accéder aux données de la grille.

Les échecs d'applications ou les autres échecs permanents (exceptions DuplicateKeyException et KeyNotFoundException) envoient immédiatement un retour et le client ne réessaie pas la transaction.

Le mode fail-fast retourne toutes les exceptions sans aucune nouvelle tentative quelle que soit l'exception.

Voici plus en détail la liste de ces exceptions :

Exceptions où le client refait des tentatives

- ReplicationVotedToRollbackTransactionException (uniquement en validation automatique)
- TargetNotAvailable
- org.omg.CORBA.SystemException
- AvailabilityException (uniquement en validation automatique)
- LockTimeoutException (uniquement en validation automatique)
- UnavailableServiceException (uniquement en validation automatique)

Autres exceptions

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

Configuration de la propriété requestRetryTimeout dans un fichier de propriétés de client

Pour définir la valeur de requestRetryTimeout dans un client, ajoutez ou modifiez la propriété dans le «Fichier de propriétés du client», à la page 205. Les propriétés du client sont définies par défaut dans le fichier `objectGridClient.properties`. La propriété requestRetryTimeout est définie en millisecondes. Une valeur supérieure à zéro indique que la demande doit être réessayée en cas de survenue d'exceptions pour lesquelles le retry est possible. Une valeur de `0` indique que les échecs ne donnent lieu à aucune nouvelle tentative dans les exceptions. Pour utiliser le comportement par défaut, supprimez la propriété ou donnez-lui une valeur de `-1`.

objectGridClient.properties

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

La valeur de requestRetryTimeout est spécifiée en millisecondes. Dans notre exemple, si la valeur est utilisée dans une instance ObjectGrid, la valeur de requestRetryTimeout sera de 30 secondes.

Définir les propriétés du client en obtenant par programmation une connexion à l'ObjectGrid

Pour définir par programmation les propriétés du client, commencez par créer un fichier de propriétés dans un `<emplacement>` approprié à votre application. Dans l'exemple suivant, le fichier des propriétés du client correspond au bout de code `objectGridClient.properties` de la section précédente. Après avoir établi une connexion à ObjectGridManager, définissez les propriétés du client en procédant comme nous allons l'indiquer. Dès lors, lorsque vous aurez une instance ObjectGrid, celle-ci aura les propriétés définies dans le fichier des propriétés du client. A chaque fois que vous serez amené à modifier ce fichier, vous devrez explicitement obtenir une nouvelle instance ObjectGrid.

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

Exemple de substitution du délai par un objet Session (validation automatique)

Pour définir dans l'objet Session pendant combien de temps effectuer de nouvelles tentatives ou pour remplacer la propriété client requestRetryTimeout, appelez la méthode setRequestRetryTimeout(long) dans l'interface Session.

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

Cette session utilise à présent pour le requestRetryTimeout une valeur de 30000 millisecondes (30 secondes), quelle que soit la valeur définie dans le fichier des propriétés du client. Pour plus d'informations sur l'interface Session, voir Utilisation des objets Session pour accéder aux données de la grille.

Configuration d'entités

Un ObjectGrid peut posséder un nombre quelconque de schémas d'entité logiques. Les entités sont définies à l'aide de classes Java annotées, d'un XML ou d'une combinaison d'un XML et de classes Java. Les entités définies sont ensuite enregistrées auprès d'un serveur eXtreme Scale et liées à des mappes de sauvegarde, des index et d'autres plug-in.

Avant de commencer

Un schéma d'entité est composé d'un ensemble d'entités et des relations entre ces entités. Voir dans Définir un schéma d'entité les explications sur la définition de schéma et la configuration d'entités.

Gestion des relations

Les langages orientés objet comme Java et les bases de données relationnelles prennent en charge les relations et les associations. Les relations diminuent la quantité d'espace de stockage utilisé grâce à l'utilisation de référence d'objet ou de clés externes.

L'utilisation de relations dans une grille oblige les données à être organisées en arborescence soumise à contraintes. Il ne doit y avoir qu'un seul type de racine dans l'arborescence et tous les enfants ne doivent être associés qu'à une seule racine. Exemple : un département pourra avoir de nombreux salariés et un salarié pourra avoir de nombreux projets. Mais un projet ne pourra avoir de nombreux salariés qui appartiennent à des départements différents. Une fois qu'une racine est définie, tous les accès à cet objet root et à ses descendants sont gérés via la racine. WebSphere eXtreme Scale utilise le code de hachage de la clé de l'objet root pour choisir une partition.

Exemple : partition = (hashCode MOD numPartitions).

Lorsque toutes les données d'une relation sont liées à une seule instance d'objet, l'instance, l'arborescence peut être située en totalité dans la même partition et il suffit d'une transaction pour y accéder de manière très efficace. Si les données embrassent plusieurs relations, plusieurs partitions seront nécessaires, ce qui implique des appels distants supplémentaires, avec le risque de goulots d'étranglement pour les performances.

Données de référence

Certaines relations incluent des données de recherche ou de référence comme CountryName, par exemple. On a là un cas très particulier où les données doivent exister dans chaque partition. Ici, les données sont accessibles à n'importe quelle clé root et les résultats retournés seront tous identiques. Des données de référence

de ce type ne doivent être utilisées que dans les cas où les données sont statiques car leur actualisation pourra s'avérer très onéreuses, devant s'effectuer dans chacune des partitions. L'API DataGrid est une technique usuellement employée pour conserver à jour les données de référence.

Coûts et avantages de la normalisation

Normaliser les données à l'aide de relations peut contribuer à réduire la quantité de mémoire utilisée par la grille puisqu'il y a moins de duplication des données à opérer. Mais, en règle générale, plus l'on ajoute de données relationnelles et moindre est leur extensibilité. En effet, lorsque les données sont regroupées, la maintenance de leurs relations devient plus onéreuse et il devient difficile de leur conserver des tailles gérables. Comme la grille partitionne les données en fonction de la clé de la racine de l'arborescence, la taille de celle-ci n'est pas prise en compte. En conséquence de quoi, si vous avez un grand nombre de relations pour une instance d'arborescence, la grille risque de devenir déséquilibrée, avec une partition détenant davantage de données que les autres.

Lorsque les données sont dénormalisées ou mises à plat sans relations hiérarchiques, les données qui, normalement, sont partagées entre deux objets sont au contraire dupliquées et chaque table peut être partitionnée de manière indépendante, ce qui donne une grille beaucoup plus équilibrée. Bien que cela augmente la quantité de mémoire utilisée, cela permet à l'application de se mettre à l'échelle puisqu'on est sûr que l'accès à une seule ligne de données donne accès à toutes les données nécessaires. C'est parfait pour les grilles qui sont essentiellement en lecture où la maintenance des données devient plus onéreuse.

Pour plus d'informations, voir Classifying XTP systems and scaling.

Gérer les relations à l'aide des API d'accès aux données

ObjectMap est l'API la plus rapide, la plus flexible et la plus granulaire de toutes les API d'accès aux données, car elle fournit une approche transactionnelle à base de sessions pour l'accès aux données présentes dans la grille des mappes. Elle permet aux clients d'utiliser des opérations CRUD (create, read, update et delete) usuelles pour gérer des paires clé/valeur d'objets dans la grille répartie.

Lorsqu'on utilise l'API ObjectMap, les relations entre les objets doivent être exprimées par l'incorporation dans l'objet parent de la clé externe de toutes les relations.

Exemple :

```
public class Department {
 Collection<String> employeeIds;
}
```

L'API EntityManager simplifie la gestion des relations en extrayant les données persistantes des objets, y compris les clés externes. Lorsque l'objet est ultérieurement récupéré dans la grille, le graphe des relations est reconstruit, comme dans l'exemple qui suit :

```
@Entity
public class Department {
 Collection<String> employees;
}
```

L'API EntityManager est très semblable aux autres technologies Java de persistance d'objet comme JPA et Hibernate, en ce qu'elle synchronise un graphe d'instances

d'objets Java gérés avec le stockage de persistance. Dans ce cas, le i est une grille eXtreme Scale grid, où chaque entité est représentée sous la forme d'une mappe et où la mappe contient les données de l'entité plutôt que les instances d'objets.

Fichier XML du descripteur de métadonnées d'entité

Le fichier du descripteur de métadonnées d'entité est un fichier XML utilisé pour définir un schéma d'entité pour WebSphere eXtreme Scale. Définissez toutes les métadonnées d'entité dans le fichier XML ou définissez les métadonnées d'entité comme annotations sur le fichier de classe Java de l'entité. Il est principalement destiné aux entités qui ne peuvent pas utiliser d'annotations Java.

Utilisez une configuration XML pour créer des métadonnées d'entité basées sur le fichier XML. Lorsqu'ils sont utilisés conjointement avec une annotation, certains des attributs définis dans la configuration XML remplacent les annotations correspondantes. Si vous remplacez un élément, ce remplacement se trouve explicitement dans les sections qui suivent. Pour un exemple de fichier XML de descripteur de métadonnées d'entité, voir «Fichier `emd.xsd`», à la page 229.

Elément id

L'élément id implique que l'attribut est une clé. Vous devez spécifier au moins un élément id. Vous pouvez spécifier plusieurs clés id à utiliser comme clé composée.

Attributs

name
Indique le nom de l'attribut. Cet attribut doit exister dans le fichier Java.

alias
Indique l'alias de l'élément. La valeur de l'alias est remplacée si elle est utilisée conjointement avec une entité annotée.

Elément basic

L'élément basic implique que l'attribut correspond à un type de primitive ou à des encapsuleurs de types de primitive :

- 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[]
- enum (Java Platform, Standard Edition Version 5)

Il n'est pas nécessaire de spécifier d'attribut basic. Les attributs de l'élément basic sont configurés automatiquement à l'aide de la réflexion.

Elément id-class

L'élément id_class spécifie une classe de clé composée, qui permet de rechercher les entités contenant des clés composées.

Attributs

class-name
Indique le nom de la classe, qui est une classe d'id, à utiliser avec l'élément id-class.

transient

L'élément transient implique que cet élément est ignoré et non traité. Il peut également être remplacé s'il est utilisé conjointement avec des entités annotées.

Attributs

name
Indique le nom de l'attribut, qui est ignoré.

version

L'élément transient implique que cet élément est ignoré et non traité. Il peut également être remplacé s'il est utilisé conjointement avec des entités annotées.

Attributs

name
Indique le nom de l'attribut, qui est ignoré.

Elément property

L'élément property permet d'ajouter des propriétés aux plug-in. Le nom de la propriété doit correspondre à une méthode set sur la classe référencée par le bean qui la contient.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : Aucun

Attributs

name
Indique le nom de la propriété. La valeur affectée à cet attribut doit correspondre à une méthode set sur la classe fournie comme attribut className sur le bean qui la contient. Par exemple, si vous affectez à l'attribut className du bean la valeur `com.ibm.MyPlugin` et que le nom de la propriété fournie est `size`, la classe `com.ibm.MyPlugin` doit contenir une méthode setSize. (Obligatoire)

type
Indique le type de la propriété. Le type est transmis à la méthode set identifiée par l'attribut name. Les valeurs valides sont les primitives Java, les équivalents java.lang et java.lang.String. Les attributs name et type doivent correspondre à une signature de méthode sur l'attribut className du bean. Par exemple, si vous spécifiez le nom `size` et le type `int`, une méthode setSize(int) doit exister sur la classe spécifiée comme attribut className pour le bean. (Obligatoire)

value
Indique la valeur de propriété. Cette valeur est convertie dans le type spécifié par l'attribut type, puis utilisée comme paramètre dans l'appel de la méthode set identifiée par les attributs name et type. La valeur de cet attribut n'est pas validée de quelque manière que ce soit. (Obligatoire)

description
Décrit la propriété (facultatif).

```
<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"
/>
```

Dans l'exemple ci-après, le fichier companyGridProperty.xml permet d'illustrer l'ajout d'un élément property à un bean. Dans cet exemple, une propriété dont le nom est maxSize et le type est int est ajoutée à un expulseur. L'expulseur com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor possède une signature de méthode qui correspond à la méthode setMaxSize(int). L'entier 499 est transmis à la méthode setMaxSize(int) sur la classe 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="The maximum size of the LRU Evictor"/>
   </bean>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

L'exemple de code suivant illustre l'approche par programme permettant d'obtenir la même configuration qu'avec le fichier companyGridProperty.xml de l'exemple précédent.

```
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 le fichier XML est utilisé à la place,
// la propriété ajoutée génère l'appel suivant :
lruEvictor.setMaxSize(449);
customerMap.setEvictor(lruEvictor);
```

Elément backingMapPluginsCollections

L'élément backingMapPluginsCollections sert de conteneur à tous les éléments backingMapPluginCollection. Dans le fichier `companyGridProperty.xml` de la section précédente, l'élément backingMapPluginCollections contient un élément backingMapPluginCollection avec l'ID `customerPlugins`.

- nombre d'occurrences : zéro à une
- élément enfant : élément backingMapPluginCollection

Elément backingMapPluginCollection

L'élément backingMapPluginCollection définit les plug-in BackingMap et est identifié par l'attribut **id**. Spécifiez l'attribut pluginCollectionRef pour référencer les plug-in. Si vous configurez plusieurs plug-in BackingMaps de manière similaire, chaque plug-in BackingMap peut faire référence au même élément backingMapPluginCollection.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : élément bean

Attributs

id Identifie la collection de plug-in de mappe de sauvegarde et est référencé par l'attribut pluginCollectionRef de l'élément backingMap. Chaque ID doit être unique. Si la valeur d'un attribut pluginCollectionRef ne correspond pas à l'ID d'un élément backingMapPluginCollection, la validation XML échoue. Plusieurs éléments backingMap peuvent faire référence à un même élément backingMapPluginCollection. (Obligatoire)

```
<backingMapPluginCollection
(1)   id="id"
/>
```

Dans l'exemple ci-après, le fichier `companyGridCollection.xml` permet d'illustrer la manière d'utiliser l'élément backingMapPluginCollection. Dans ce fichier, la mappe de sauvegarde Customer utilise la collection de plug-in de mappe de sauvegarde customerPlugins pour se configurer avec un expulseur LRU. Les mappes de sauvegarde Item et OrderLine font référence à la collection de plug-in de mappe de sauvegarde collection2. Ces mappes de sauvegarde possède chacune un ensemble d'expulseurs LFU.

```
<?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>
```

L'exemple de code suivant illustre l'approche par programme permettant d'obtenir la même configuration qu'avec le fichier `companyGridCollection.xml` de l'exemple précédent.

```
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");
```

Elément querySchema

L'élément querySchema définit les relations entre les mappes de sauvegarde et identifie le type d'objet dans chaque mappe. Ces informations sont utilisées par la requête d'objet pour convertir les chaînes du langage de requête en appels d'accès à la mappe. Pour plus d'informations, reportez-vous aux détails sur la définition d'un schéma ObjectQuery, dans le *Guide de programmation*.

- nombre d'occurrences : zéro à une
- élément enfant : élément mapSchemas, élément relationships

Elément mapSchemas

Chaque élément querySchema possède un élément mapSchemas qui contient un ou plusieurs éléments mapSchema.

- nombre d'occurrences : Une
- élément enfant : élément mapSchema

Elément mapSchema

Un élément mapSchema définit le type d'objet stocké dans une mappe de sauvegarde et les instructions d'accès aux données.

- nombre d'occurrences : une ou plusieurs
- élément enfant : aucun

Attributs

mapName
Indique le nom de la mappe de sauvegarde à ajouter au schéma. (Obligatoire)

valueClass
Indique le type d'objet stocké dans la portion valeur de la mappe de sauvegarde. (Obligatoire)

primaryKeyField
Indique le nom de l'attribut de clé primaire dans l'attribut valueClass. La clé primaire doit également être stockée dans la portion clé de la mappe de sauvegarde (facultatif).

accessType
Identifie la manière dont le moteur de requête introspecte les données persistantes des instances d'objet valueClass et y accède. Si vous spécifiez la valeur `FIELD`, les champs de classes sont introspectés et ajoutés au schéma. Si la

valeur est PROPERTY, les attributs associés aux méthodes get et is sont utilisés. La valeur par défaut est PROPERTY (facultatif).

```
<mapSchema
(1)   mapName="backingMapName"
(2)   valueClass="com.mycompany.OrderBean"
(3)   primaryKeyField="orderId"
(4)   accessType="PROPERTY" | "FIELD"
/>
```

Dans l'exemple ci-après, le fichier companyGridQuerySchemaAttr.xml permet d'illustrer un exemple de configuration mapSchema.

```
<?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>
```

L'exemple de code suivant illustre l'approche par programme permettant d'obtenir la même configuration qu'avec le fichier companyGridQuerySchemaAttr.xml de l'exemple précédent.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Définissez le schéma
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);
```

Elément relationships

Chaque élément querySchema possède zéro ou un élément relationships qui contient un ou plusieurs éléments relationship.

- nombre d'occurrences : zéro ou une
- élément enfant : élément relationship

Elément relationship

Un élément relationship définit la relation entre deux mappes de sauvegarde et les attributs de l'attribut valueClass qui associent la relation.

- nombre d'occurrences : une ou plusieurs
- élément enfant : aucun

Attributs

source
Indique le nom de la classe de valeur du côté source d'une relation. (Obligatoire)

target
Indique le nom de la classe de valeur du côté cible d'une relation. (Obligatoire)

relationField
Indique le nom de l'attribut dans la classe de valeur source qui fait référence à la cible. (Obligatoire)

invRelationField
Indique le nom de l'attribut dans la classe de valeur cible qui fait référence à la source. Si cet attribut n'est pas spécifié, la relation est unidirectionnelle (facultatif).

```
<mapSchema
(1)   source="com.mycompany.OrderBean"
(2)   target="com.mycompany.CustomerBean"
(3)   relationField="customer"
(4)   invRelationField="orders"
/>
```

Dans l'exemple ci-après, le fichier `companyGridQuerySchemaWithRelationshipAttr.xml` permet d'illustrer un exemple de configuration mapSchema qui inclut une relation bidirectionnelle.

```
<?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>
```

L'exemple de code suivant illustre l'approche par programme permettant d'obtenir la même configuration qu'avec le fichier `companyGridQuerySchemaWithRelationshipAttr.xml` de l'exemple précédent.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Définissez le schéma
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);
```

Elément timeBasedDBUpdate

Un élément timeBasedDBUpdate définit une configuration pour un programme de mise à jour de base de données en fonction de la date/heure. Un élement timeBasedDBUpdate contient des informations sur la fréquence d'extraction des enregistrements nouvellement insérés et mis à jour à partir de la base de données à l'aide de l'API JPA (Java Persistence API) et sur la manière de mettre à jour les données dans les mappes ObjectGrid correspondantes.

- nombre d'occurrences : zéro ou une
- élément enfant : aucun

Attributs

entityClass
Indique le nom de classe d'entités utilisé pour interagir avec le fournisseur JPA. Ce nom est utilisé pour extraire les entités JPA à l'aide de requêtes d'entité. (Obligatoire)

persistenceUnitName
Indique le nom de l'unité de persistance JPA pour la création d'une fabrique de gestionnaire d'entités JPA. La valeur par défaut est le nom de la première unité de persistance définie dans le fichier `persistence.xml` (facultatif).

mode
Indique le mode de mise à jour de base de données en fonction de la date/heure. Par défaut, le mode de mise à jour de base de données en fonction de la date/heure est INVALIDATE_ONLY. Un type INVALIDATE_ONLY indique d'invalider les entrées de la mappe ObjectGrid si les enregistrements correspondants de la base de données ont été modifiés. Un type UPDATE_ONLY indique de remplacer les entrées existantes dans la mappe ObjectGrid par les dernières valeurs de la base de données. Toutefois, tous les enregistrements nouvellement insérés dans la base de données sont ignorés. Un type INSERT_UPDATE indique de remplacer les entrées existantes dans la mappe ObjectGrid par les dernières valeurs de la base de données. En outre, tous les enregistrements nouvellement insérés dans la base de données sont insérées dans la mappe ObjectGrid (facultatif).

timestampField
Indique le nom du champ d'horodatage. Valeur de champ d'horodatage permettant d'identifier l'heure ou la séquence de la dernière mise à jour d'un enregistrement dorsal de base de données (facultatif).

jpaPropertyFactory
Identifie le bean Spring ou le nom de classe de l'implémentation

JPAPropertyFactory. L'interface com.ibm.websphere.objectgrid.jpa.JPAPropertyFactory est utilisée pour intégrer la mappe des propriétés de persistance afin de remplacer les propriétés JPA par défaut. Utilisez les beans de l'infrastructure Spring si des attributs supplémentaires doivent être définis sur l'instance JPAPropertyFactory. Pour plus d'informations, voir Présentation générale de l'intégration à Spring (facultatif).

```
<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"
/>
```

Elément streamQuerySet

L'élément streamQuerySet est l'élément de niveau supérieur de la définition d'un ensemble de requêtes de flux.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : élément stream, élément view

Elément stream

L'élément stream représente un flux vers le moteur de requête de flux. Chaque attribut de l'élément stream correspond à une méthode de l'interface StreamMetadata.

- nombre d'occurrences : une à plusieurs
- élément enfant : élément basic

Attributs

name
Indique le nom du flux. La validation échoue si cet attribut n'est pas spécifié. (Obligatoire)

valueClass
Indique le type de classe de la valeur stockée dans la mappe d'objets de flux. Le type de classe permet de convertir l'objet en événements de flux et de générer une instruction SQL si cette dernière n'est pas fournie. (Obligatoire)

sql
Indique l'instruction SQL du flux. Si cette propriété n'est pas fournie, un SQL de flux est généré en reflétant les attributs ou les méthodes d'accès sur l'attribut valueClass ou en utilisant les attributs de nuplet des métadonnées d'entité (facultatif).

access
Indique le type d'accès aux attributs de la classe de valeur. Si vous spécifiez la valeur`FIELD`, les attributs sont directement extraits des champs à l'aide de la réflexion Java. Sinon, les méthodes d'accès sont utilisées pour lire les attributs. La valeur par défaut est `PROPERTY` (facultatif).

```
<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"
/>
```

Elément view

L'élément view représente une vue de requête de flux. Chaque élément stream correspond à une méthode de l'interface ViewMetadata.

- nombre d'occurrences : une à plusieurs
- élément enfant : élément basic, élément id

Attributs

name
Indique le nom de la vue. La validation échoue si cet attribut n'est pas spécifié. (Obligatoire)

sql
Indique le SQL du flux, qui définit la transformation de la vue. La validation échoue si cet attribut n'est pas spécifié. (Obligatoire)

valueClass
Indique le type de classe de la valeur stockée dans cette vue de la mappe d'objets. Le type de classe permet de convertir les événements de vue dans le format de nuplet approprié compatible avec ce type de classe. Si le type de classe n'est pas fourni, un format par défaut qui suit les définitions de colonne dans le langage SPTSQL (Stream Processing Technology Structured Query Language) est utilisé. Si les métadonnées d'une entité sont définies pour cette mappe de vues, cet attribut ne doit pas être utilisé. Les métadonnées sont utilisées à la place (facultatif).

access
Indique le type d'accès aux attributs de la classe de valeur. Si vous spécifiez le type d'accès `FIELD`, les valeurs des colonnes reçoivent directement les valeurs des champs à l'aide de la réflexion Java. Sinon, les méthodes d'accès sont utilisées pour définir les attributs. La valeur par défaut est `PROPERTY` (facultatif).

```
<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"
/>
```

Elément basic

L'élément basic permet de définir un mappage entre le nom d'attribut de la classe de valeur ou les métadonnées d'entité et la colonne définie dans le langage SPTSQL.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

```
<basic
(1)   name="attributeName"
(2)   column="columnName"
/>
```

Elément id

L'élément id est utilisé pour un mappage d'attributs de clé.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

```
<id
(1)   name="idName"
(2)   column="columnName"
/>
```

Dans l'exemple ci-après, le fichier `StreamQueryApp2.xml` permet d'illustrer la configuration des attributs d'un élément streamQuerySet. L'ensemble de requêtes de flux _stockQuoteSQS_ contient un flux et une vue. Ce flux et cette vue définissent respectivement son nom, sa classe de valeur, le SQL et le type d'accès. Le flux définit également un élément basic, qui spécifie que l'attribut volume de la classe StockQuote est mappé à la colonne SQL transactionvolume définie dans l'instruction 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>
```

Fichier `emd.xsd`

Utilisez la définition de schéma XML de métadonnées d'entité pour créer un fichier XML de descripteur et définir un schéma d'entité pour WebSphere eXtreme Scale.

Pour les descriptions de chaque élément et attribut du fichier `emd.xsd`, voir la rubrique «Fichier XML du descripteur de métadonnées d'entité», à la page 219.

Fichier `emd.xsd`

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:emd="http://ibm.com/ws/projector/config/emd"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://ibm.com/ws/projector/config/emd"
    elementFormDefault="qualified" attributeFormDefault="unqualified"
    version="1.0">
```

```
<!-- ************************************************** -->
<xsd:element name="entity-mappings">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="description" type="xsd:string" minOccurs="0"/>
            <xsd:element name="entity" type="emd:entity" minOccurs="1" maxOccurs="unbounded" />
        </xsd:sequence>
    </xsd:complexType>
    <xsd:unique name="uniqueEntityClassName">
        <xsd:selector xpath="emd:entity" />
        <xsd:field xpath="@class-name" />
    </xsd:unique>
</xsd:element>

<!-- ************************************************** -->
<xsd:complexType name="entity">
    <xsd:sequence>
        <xsd:element name="description" type="xsd:string" minOccurs="0"/>
        <xsd:element name="id-class" type="emd:id-class" minOccurs="0" />
        <xsd:element name="attributes" type="emd:attributes" minOccurs="0" />
        <xsd:element name="entity-listeners" type="emd:entity-listeners" minOccurs="0" />
        <xsd:element name="pre-persist" type="emd:pre-persist" minOccurs="0" />
        <xsd:element name="post-persist" type="emd:post-persist" minOccurs="0" />
        <xsd:element name="pre-remove" type="emd:pre-remove" minOccurs="0" />
        <xsd:element name="post-remove" type="emd:post-remove" minOccurs="0" />
        <xsd:element name="pre-invalidate" type="emd:pre-invalidate" minOccurs="0" />
        <xsd:element name="post-invalidate" type="emd:post-invalidate" minOccurs="0" />
        <xsd:element name="pre-update" type="emd:pre-update" minOccurs="0" />
        <xsd:element name="post-update" type="emd:post-update" minOccurs="0" />
        <xsd:element name="post-load" type="emd:post-load" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="class-name" type="xsd:string" use="required" />
    <xsd:attribute name="access" type="emd:access-type" />
    <xsd:attribute name="schemaRoot" type="xsd:boolean" />
</xsd:complexType>

<!-- ************************************************** -->
<xsd:complexType name="attributes">
    <xsd:sequence>
        <xsd:choice>
            <xsd:element name="id" type="emd:id" minOccurs="0" maxOccurs="unbounded" />
        </xsd:choice>
        <xsd:element name="basic" type="emd:basic" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="version" type="emd:version" minOccurs="0" maxOccurs="unbounded"/>
        <xsd:element name="many-to-one" type="emd:many-to-one" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="one-to-many" type="emd:one-to-many" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="one-to-one" type="emd:one-to-one" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="many-to-many" type="emd:many-to-many" minOccurs="0" maxOccurs="unbounded" />
        <xsd:element name="transient" type="emd:transient" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
</xsd:complexType>

<!-- ************************************************** -->
<xsd:simpleType name="access-type">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="PROPERTY" />
        <xsd:enumeration value="FIELD" />
    </xsd:restriction>
</xsd:simpleType>

<!-- ************************************************** -->
<xsd:complexType name="id-class">
    <xsd:attribute name="class-name" type="xsd:string" use="required" />
</xsd:complexType>

<!-- ************************************************** -->
<xsd:complexType name="id">
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="type" type="xsd:string" />
    <xsd:attribute name="alias" type="xsd:string" use="optional" />
</xsd:complexType>

<!-- ************************************************** -->
<xsd:complexType name="transient">
    <xsd:attribute name="name" type="xsd:string" use="required" />
</xsd:complexType>

<!-- ************************************************** -->
<xsd:complexType name="basic">
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="alias" type="xsd:string" />
    <xsd:attribute name="type" type="xsd:string" />
    <xsd:attribute name="fetch" type="emd:fetch-type" />
</xsd:complexType>

<!-- ************************************************** -->
<xsd:simpleType name="fetch-type">
    <xsd:restriction base="xsd:token">
        <xsd:enumeration value="LAZY" />
```

```
        <xsd:enumeration value="EAGER" />
    </xsd:restriction>
</xsd:simpleType>

<!-- ************************************************** -->
<xsd:complexType name="many-to-one">
    <xsd:sequence>
        <xsd:element name="cascade" type="emd:cascade-type" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="alias" type="xsd:string" />
    <xsd:attribute name="target-entity" type="xsd:string" />
    <xsd:attribute name="fetch" type="emd:fetch-type" />
    <xsd:attribute name="id" type="xsd:boolean" />
</xsd:complexType>
<!-- ************************************************** -->
<xsd:complexType name="one-to-one">
    <xsd:sequence>
        <xsd:element name="cascade" type="emd:cascade-type" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="alias" type="xsd:string" />
    <xsd:attribute name="target-entity" type="xsd:string" />
    <xsd:attribute name="fetch" type="emd:fetch-type" />
    <xsd:attribute name="mapped-by" type="xsd:string" />
    <xsd:attribute name="id" type="xsd:boolean" />
</xsd:complexType>
<!-- ************************************************** -->
<xsd:complexType name="one-to-many">
    <xsd:sequence>
        <xsd:element name="order-by" type="emd:order-by" minOccurs="0" />
        <xsd:element name="cascade" type="emd:cascade-type" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="alias" type="xsd:string" />
    <xsd:attribute name="target-entity" type="xsd:string" />
    <xsd:attribute name="fetch" type="emd:fetch-type" />
    <xsd:attribute name="mapped-by" type="xsd:string" />
</xsd:complexType>

<!-- ************************************************** -->
<xsd:complexType name="many-to-many">
    <xsd:sequence>
        <xsd:element name="order-by" type="emd:order-by" minOccurs="0" />
        <xsd:element name="cascade" type="emd:cascade-type" minOccurs="0" />
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required" />
    <xsd:attribute name="alias" type="xsd:string" />
    <xsd:attribute name="target-entity" type="xsd:string" />
    <xsd:attribute name="fetch" type="emd:fetch-type" />
    <xsd:attribute name="mapped-by" type="xsd:string" />
</xsd:complexType>

<!-- ************************************************** -->
<xsd:simpleType name="order-by">
    <xsd:restriction base="xsd:string" />
</xsd:simpleType>

<!-- ************************************************** -->
<xsd:complexType name="cascade-type">
    <xsd:sequence>
        <xsd:element name="cascade-all" type="emd:emptyType" minOccurs="0" />
        <xsd:element name="cascade-persist" type="emd:emptyType" minOccurs="0" />
        <xsd:element name="cascade-remove" type="emd:emptyType" minOccurs="0" />
        <xsd:element name="cascade-invalidate" type="emd:emptyType" minOccurs="0" />
        <xsd:element name="cascade-merge" type="emd:emptyType" minOccurs="0" />
        <xsd:element name="cascade-refresh" type="emd:emptyType" minOccurs="0" />
    </xsd:sequence>
</xsd:complexType>

<!-- ************************************************** -->
<xsd:complexType name="emptyType" />

<!-- ************************************************** -->
<xsd:complexType name="version">
      <xsd:attribute name="name" type="xsd:string" use="required"/>
      <xsd:attribute name="alias" type="xsd:string" />
      <xsd:attribute name="type" type="xsd:string" />
</xsd:complexType>

<!-- ************************************************** -->

<xsd:complexType name="entity-listeners">
    <xsd:sequence>
        <xsd:element name="entity-listener" type="emd:entity-listener" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
</xsd:complexType>

<!-- ************************************************** -->
<xsd:complexType name="entity-listener">
    <xsd:sequence>
```

```
            <xsd:element name="pre-persist" type="emd:pre-persist" minOccurs="0" />
            <xsd:element name="post-persist" type="emd:post-persist" minOccurs="0" />
            <xsd:element name="pre-remove" type="emd:pre-remove" minOccurs="0" />
            <xsd:element name="post-remove" type="emd:post-remove" minOccurs="0" />
            <xsd:element name="pre-invalidate" type="emd:pre-invalidate" minOccurs="0" />
            <xsd:element name="post-invalidate" type="emd:post-invalidate" minOccurs="0" />
            <xsd:element name="pre-update" type="emd:pre-update" minOccurs="0" />
            <xsd:element name="post-update" type="emd:post-update" minOccurs="0" />
            <xsd:element name="post-load" type="emd:post-load" minOccurs="0" />
        </xsd:sequence>
        <xsd:attribute name="class-name" type="xsd:string" use="required" />
    </xsd:complexType>

    <!-- ************************************************** -->
    <xsd:complexType name="pre-persist">
        <xsd:attribute name="method-name" type="xsd:string" use="required" />
    </xsd:complexType>

    <!-- ************************************************** -->
    <xsd:complexType name="post-persist">
        <xsd:attribute name="method-name" type="xsd:string" use="required" />
    </xsd:complexType>

    <!-- ************************************************** -->
    <xsd:complexType name="pre-remove">
        <xsd:attribute name="method-name" type="xsd:string" use="required" />
    </xsd:complexType>

    <!-- ************************************************** -->
    <xsd:complexType name="post-remove">
        <xsd:attribute name="method-name" type="xsd:string" use="required" />
    </xsd:complexType>

    <!-- ************************************************** -->
    <xsd:complexType name="pre-invalidate">
        <xsd:attribute name="method-name" type="xsd:string" use="required" />
    </xsd:complexType>

    <!-- ************************************************** -->
    <xsd:complexType name="post-invalidate">
        <xsd:attribute name="method-name" type="xsd:string" use="required" />
    </xsd:complexType>

    <!-- ************************************************** -->
    <xsd:complexType name="pre-update">
        <xsd:attribute name="method-name" type="xsd:string" use="required" />
    </xsd:complexType>

    <!-- ************************************************** -->
    <xsd:complexType name="post-update">
        <xsd:attribute name="method-name" type="xsd:string" use="required" />
    </xsd:complexType>

    <!-- ************************************************** -->
    <xsd:complexType name="post-load">
        <xsd:attribute name="method-name" type="xsd:string" use="required" />
    </xsd:complexType>

</xsd:schema>
```

Configuration de l'intégration du cache

WebSphere eXtreme Scale peut s'intégrer aux autres produits de mise en cache. JPA peut être utilisé entre WebSphere eXtreme Scale et la base de données pour intégrer les modifications en tant que chargeur. Vous pouvez aussi utiliser le fournisseur de cache dynamique WebSphere eXtreme Scale pour connecter WebSphere eXtreme Scale au composant de cache dynamique WebSphere Application Server. Autre extension de WebSphere Application Server : le gestionnaire de sessions HTTP WebSphere eXtreme Scale, qui permet la mise en cache des sessions HTTP.

Intégration du cache : mise en cache des JPA et des sessions et mise en cache dynamique

L'élément crucial qui donne à WebSphere eXtreme Scale son extrême adaptabilité et fiabilité réside dans sa conception des caches, qui lui permet d'optimiser la persistance et la collecte des données dans quasiment n'importe quel environnement de déploiement.

Configuration des chargeurs JPA

Un chargeur Java Persistence API (JPA) est une implémentation de plug-in qui utilise JPA pour interagir avec la base de données.

Avant de commencer

- Vous devez disposer d'une implémentation JPA, comme Hibernate ou OpenJPA.
- Votre base de données peut correspondre à tout programme d'arrière plan prise en charge par le fournisseur JPA choisi.
- Vous pouvez utiliser le plug-in JPALoader lorsque vous stockez des données à l'aide de l'API ObjectMap. Utilisez le plug-in JPAEntityLoader lorsque vous stockez des données à l'aide de l'API EntityManager.

Pourquoi et quand exécuter cette tâche

Pour plus d'information sur le fonctionnement du chargeur Java Persistence API (JPA), voir les informations du document *Présentation du produit*.

Procédure

1. Configurez les paramètres requis par JPA pour interagir avec une base de données.

   Les paramètres ci-après sont requis. Ces paramètres sont configurés dans le bean JPALoader ou JPAEntityLoader et le bean JPATxCallback.

   - **persistenceUnitName** : Indique le nom de l'unité de persistance. Ce paramètre est requis à deux titres : pour créer une fabrique de gestionnaire d'entités JPA et pour rechercher les métadonnées d'entité JPA dans le fichier `persistence.xml`. Cet attribut est défini sur le bean JPATxCallback.
   - **JPAPropertyFactory** : Indique la fabrique permettant de créer une mappe de propriétés de persistance pour remplacer les propriétés de persistance par défaut. Cet attribut est défini sur le bean JPATxCallback. Pour définir cet attribut, une configuration de style Spring est requise.
   - **entityClassName** : Indique le nom de classe d'entité requis pour utiliser les méthodes JPA (par exemple, EntityManager.persist, EntityManager.find, etc.). La chargeur JPA requiert ce paramètre, mais ce paramètre est facultatif pour **JPAEntityLoader**. Dans le cas de JPAEntityLoader, si un paramètre **entityClassName** n'est pas configuré, la classe d'entités configurée dans la mappe d'entités d'ObjectGrid est utilisée. Vous devez utiliser le même nom de classe pour le gestionnaire d'entités eXtreme Scale et le fournisseur JPA. Cet attribut est défini sur le bean JPALoader ou JPAEntityLoader.
   - **preloadPartition** : Indique la partition à partir de laquelle le préchargement de la mappe démarre. Si la partition de préchargement est inférieure à zéro ou supérieure au nombre total de partitions moins 1, le préchargement de la mappe n'est pas démarré. La valeur par défaut est -1, ce qui signifie que le préchargement ne démarre pas par défaut. Cet attribut est défini sur le bean JPALoader ou JPAEntityLoader.

   En plus des quatre paramètres JPA à configurer dans eXtreme Scale, les métadonnées JPA sont utilisées pour extraire la clé des entités JPA. Les métadonnées JPA peuvent être configurées comme annotation ou comme fichier `orm.xml` spécifié dans le fichier `persistence.xml`. Elles ne font pas partie de la configuration d'eXtreme Scale.
2. Configurez les fichiers XML de la configuration JPA.

   Pour configurer un chargeur JPA ou un chargeur d'entité JPA, voir les informations sur les plug-in de chargeur dans le *Guide de programmation*.

Configurez un rappel de transaction JPATxCallback avec la configuration du chargeur. L'exemple suivant illustre un fichier de descripteur XML ObjectGrid (objectgrid.xml), pour lequel JPAEntityLoader et JPATxCallback sont configurés :

configuration d'un chargeur avec rappel - Exemple 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 vous souhaitez configurer une fabrique de propriétés JPA, vous devez utiliser une configuration de style Spring. Vous trouverez ci-après un exemple de fichier de configuration XML, JPAEM_spring.xml, qui configure un bean Spring à utiliser pour les configurations eXtreme Scale.

configuration d'un chargeur avec une fabrique de propriétés JPA - Exemple 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>
```

Le fichier de configuration XML Objectgrid.xml est présenté ci-après. Notez que le nom ObjectGrid est JPAEM, qui correspond au nom ObjectGrid dans le fichier de configuration JPAEM_spring.xml de Spring.

Configuration du chargeur JPAEM - Exemple 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>
```

Une entité peut être annotée avec les annotations JPA et les annotations du gestionnaire d'entités d'eXtreme Scale. Chaque annotation possède un équivalent XML qui peut être utilisé. eXtreme Scale a donc ajouté l'espace de noms Spring. Vous pouvez également les configurer à l'aide de la prise en charge de l'espace de noms Spring.

Configuration d'un programme de mise à jour de données JPA en fonction de la date/heure

Vous pouvez configurer une mise à jour de base de données en fonction de la date/heure à l'aide d'une configuration XML eXtreme Scale locale ou répartie. Vous pouvez également configurer une configuration locale à l'aide d'un programme.

Pourquoi et quand exécuter cette tâche

Pour plus d'information sur le fonctionnement du programme de mise à jour de données Java Persistence API (JPA) en fonction de la date/heure, voir les informations du *Guide de programmation*.

Procédure

Créez une configuration timeBasedDBUpdate.

- **Avec un fichier XML :**

  L'exemple suivant illustre un fichier `objectgrid.xml` qui contient une configuration timeBasedDBUpdate :

```
Programme de mise à jour
JPA en fonction de la date/heure - Exemple 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>
```

  Dans cet exemple, la mappe "user" est configurée avec une mise à jour de base de données en fonction de la date/heure. Le mode de mise à jour de base de données est INVALIDATE_ONLY et le champ d'horodatage possède la valeur rowChgTs.

  Si l'ObjectGrid réparti "changeOG" est démarré sur le serveur conteneur, une unité d'exécution de mise à jour de base de données en fonction de la date/heure est automatiquement démarrée dans la partition 0.

- **A l'aide d'un programme :**

  Si vous créez un ObjectGrid local, vous pouvez également créer un objet TimeBasedDBUpdateConfig et le définir sur l'instance BackingMap :

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

Pour plus d'informations sur la définition d'un objet sur l'instance BackingMap, voir les informations sur l'interface BackingMap dans la documentation de l'API.

Vous pouvez également annoter le champ d'horodatage dans la classe d'entité à l'aide de l'annotation com.ibm.websphere.objectgrid.jpa.dbupdate.annotation.Timestamp. En configurant la valeur dans la classe, vous n'avez pas besoin de configurer le champ d'horodatage dans la configuration XML.

Que faire ensuite

Démarrez le programme de mise à jour de base de données JPA en fonction de la date/heure. Pour plus d'informations, voir les informations sur le démarrage du programme de mise à jour en fonction de la date/heure, dans le *Guide de programmation*.

Configuration de plug-in de cache JPA

WebSphere eXtreme Scale inclut des plug-in de cache de niveau 2 pour les fournisseurs Java Persistence API (JPA) OpenJPA et Hibernate.

Propriétés de configuration du cache JPA d'ObjectGrid

Vous pouvez configurer le plug-in du cache JPA avec les propriétés ci-après, qui sont toutes facultatives.

ObjectGridName

Indique le nom unique d'ObjectGrid. La valeur par défaut est le nom d'unité de persistance défini. Si le nom de l'unité de persistance n'est pas disponible auprès du fournisseur, un nom généré est utilisé.

ObjectGridType

Indique le type d'ObjectGrid.

Valeurs admises :

- **EMBEDDED** : Type de configuration par défaut et recommandé. Ses valeurs par défaut sont : `NumberOfPartitions=1`, `ReplicaMode=SYNC`, `ReplicaReadEnabled=true` et `MaxNumberOfReplicas=47`. Utilisez le paramètre **ReplicaMode** pour définir le mode de réplication et le paramètre **MaxNumberOfReplicas** pour définir le nombre maximal de fragments répliques. Si un système possède plus de 47 machines virtuelles Java, spécifiez pour **MaxNumberOfReplicas** une valeur égale au nombre de machines virtuelles Java.
- **EMBEDDED_PARTITION** : Type à utiliser si le système doit mettre en cache une quantité de données importante sur un système réparti. Le nombre de partitions par défaut est `47` avec le mode de réplique `NONE`. Sur un petit système qui ne possède que quelques machines virtuelles Java, spécifiez pour **NumberOfPartitions** une valeur égale ou inférieure au nombre de machines virtuelles Java. Vous pouvez spécifier les valeurs **ReplicaMode**, **NumberOfPartitions** et **ReplicaReadEnabled** pour optimiser le système.
- **REMOTE** : Le cache tente de se connecter à un ObjectGrid réparti éloigné à partir du service de catalogue.

NumberOfPartitions

Valeurs valides : Valeurs supérieures ou égales à 1Indique le nombre de partitions à utiliser pour le cache. Cette propriété s'applique si la valeur

d'ObjectGridType est EMBEDDED_PARTITION. La valeur par défaut est 47. Pour le type EMBEDDED, la valeur **NumberOfPartitions** est toujours 1.

ReplicaMode

Valeurs valides : SYNC/ASYNC/NONEIndique la méthode utilisée pour copier le cache vers les fragments répliques. Cette propriété s'applique si la valeur d'ObjectGridType est EMBEDDED ou EMBEDDED_PARTITION. La valeur par défaut est NONE pour le type EMBEDDED_PARTITION et SYNC pour le type EMBEDDED. Si la valeur de **ReplicaMode** est NONE pour le type de grille d'objets EMBEDDED, le type EMBEDDED utilise la valeur SYNC pour **ReplicaMode**.

ReplicaReadEnabled

Valeurs valides : TRUE ou FALSESi cette propriété est activée, les clients lisent les valeurs à partir des fragments répliques. Cette propriété s'applique au type EMBEDDED_PARTITION. La valeur par défaut est FALSE pour le type EMBEDDED_PARTITION. Le type EMBEDDED affecte toujours à la propriété **ReplicaReadEnabled** la valeur TRUE.

MaxUsedMemory

Valeurs valides : TRUE ou FALSEActive l'expulsion des entrées du cache si la mémoire est soumise à des contraintes. La valeur par défaut est TRUE et les données sont expulsées lorsque le seuil d'utilisation des segments de mémoire de la machine virtuelle Java dépasse 70 pourcent. Vous pouvez modifier le seuil d'utilisation des segments de mémoire de la machine virtuelle Java par défaut en définissant la propriété memoryThresholdPercentage dans le fichier objectGridServer.properties et en plaçant ce fichier dans le chemin d'accès aux classes. Pour plus d'informations sur les expulseurs, voir les informations sur les expulseurs dans le *Présentation du produit*. Pour plus d'informations sur le fichier de propriétés du serveur, voir le *Guide d'administration*.

MaxNumberOfReplicas

Valeurs valides : Valeurs supérieures ou égales à 1Indique le nombre maximal de fragments répliques à utiliser pour le cache. Cette valeur ne s'applique qu'au type EMBEDDED. Ce nombre doit être égal ou supérieur au nombre de machines virtuelles Java d'un système. La valeur par défaut est 47.

Les propriétés NumberOfPartitions, ReplicaMode, ReplicaReadEnabled, et MaxNumberOfReplicas sont des facteurs de déploiement d'ObjectGrid. Les propriétés NumberOfPartitions, ReplicaMode et ReplicaReadEnabled s'appliquent au type EMBEDDED_PARTITION. Les propriétés ReplicaMode et MaxNumberOfReplicas s'appliquent au type EMBEDDED.

Considérations liés à EMBEDDED et EMBEDDED_PARTITION

Les types d'ObjectGrid imbriqués utilisent les propriétés de configuration précédemment décrites pour configurer et déployer un ensemble de serveurs conteneurs ObjectGrid et un service de catalogue si nécessaire. Le cycle de vie des conteneurs est associé à l'application JPA et regroupé dans le chemin d'accès aux classes de l'application. Lorsqu'une application est démarrée, le plug-in détecte automatiquement un service de catalogue ou en démarre un, démarre un conteneur et se connecte au service de catalogue. Le plug-in communique alors avec le conteneur ObjectGrid et ses homologues exécutés dans d'autres processus de serveur d'applications à l'aide de la connexion client.

Chaque entité JPA possède une mappe de sauvegarde indépendante affectée à l'aide du nom de classe de l'entité. Chaque mappe de sauvegarde possède les attributs ci-après.

- `readOnly="false"`
- `copyKey="false"`
- `lockStrategy="NONE"`
- `copyMode="NO_COPY"`

Remarque : Si vous utilisez le type de grille d'objets `EMBEDDED` ou `EMBEDDED_PARTITION` dans un environnement Java SE, utilisez la méthode System.exit(0) à la fin du programme pour arrêter le serveur eXtreme Scale imbriqué. Sinon, le programme semble ne plus répondre.

Valeurs par défaut d'ObjectGridType

La valeur ObjectGridType indique la topologie dans laquelle le cache d'ObjectGrid est déployé. Le type par défaut et le plus performant est EMBEDDED. Les sections ci-après décrivent les propriétés par défaut de chacune des valeurs ObjectGridType.

Valeurs par défaut de la topologie de cache JPA d'ObjectGrid EMBEDDED

Lorsque vous utilisez le type d'ObjectGrid `EMBEDDED`, les valeurs de propriété par défaut suivantes sont utilisées si vous ne spécifiez pas de valeurs dans la configuration :

- **ObjectGridName** : nom de l'unité de persistance
- **ObjectGridType** : `EMBEDDED`
- **NumberOfPartitions** : 1 (ne peut pas être modifiée si le type d'ObjectGrid est `EMBEDDED`)
- **ReplicaMode** : `SYNC`
- **ReplicaReadEnabled** : `TRUE` (ne peut pas être modifiée si le type d'ObjectGrid est `EMBEDDED`)
- **MaxUsedMemory** : `TRUE`
- **MaxNumberOfReplicas** : 47 (doit être inférieur ou égal au nombre de machines virtuelles Java sur un système réparti)

Vous devez spécifier une valeur ObjectGridName unique pour éviter les conflits de nom. La valeur MaxNumberOfReplicas doit être égale ou supérieure au nombre total de machines virtuelles Java sur le système.

Topologie de cache d'ObjectGrid REMOTE

Le type d'ObjectGrid REMOTE ne nécessite pas de paramètres de propriété car l'ObjectGrid et la règle de déploiement sont définis distinctement de l'application JPA. Le plug-in de cache JPA se connecte à distance à un ObjectGrid éloigné existant.

Toute interaction avec la grille d'objets étant éloignée, cette topologie offre les moins bonnes performances parmi tous les types de grille d'objets.

Remarques sur le service de catalogue et configuration

Si vous utilisez une topologie `EMBEDDED` ou `EMBEDDED_PARTITION`, le plug-in de cache JPA démarre automatiquement un service de catalogue dans l'un des processus d'application si nécessaire. Dans un environnement de production, vous devez créer un domaine de services de catalogue. Pour plus d'informations sur la définition d'un service de catalogue, voir .les informations sur le service de catalogue à haute disponibilité dans le *Présentation du produit*

Lors d'une exécution à l'intérieur d'un processus WebSphere Application Server, le plug-in de cache JPA se connecte automatiquement au service de catalogue (ou au domaine de services de catalogue) qui est défini pour la cellule WebSphere Application Server. Pour savoir comment définir un domaine de services de catalogue, voir dans le *Guide d'administration* les explications sur le démarrage du service de catalogue.

Si vous n'exécutez pas vos serveurs dans un processus WebSphere Application Server, les hôtes et les ports du domaine de services de catalogue sont spécifiés à l'aide du fichier de propriétés `objectGridServer.properties`. Ce fichier doit être stocké dans le chemin d'accès aux classes de l'application et la propriété **catalogServiceEndPoints** doit être définie. La grille de service de catalogue est démarrée indépendamment des processus d'application et doit être démarrée avant les processus d'application.

Le format du fichier `objectGridServer.properties` est le suivant :

```
catalogServiceEndPoints=<hostname1>:<port1>,<hostname2>:<port2>
```

Plug-in de cache JPA

WebSphere eXtreme Scale inclut des plug-in de mémoire cache de niveau 2 pour les fournisseurs OpenJPA et Hibernate Java Persistence API (JPA).

L'utilisation d'eXtreme Scale en tant que fournisseur de cache de niveau 2 améliore les performances lors de la lecture et de l'interrogation des données et réduit la charge pesant sur la base de données. WebSphere eXtreme Scale présente plusieurs avantages par rapport aux implémentations de cache pré-intégrées car le cache est automatiquement répliqué entre tous les processus. Lorsqu'un client met une valeur en cache, tous les autres clients peuvent utiliser la valeur mise en cache en local.

Avec les plug-in OpenJPA et Hibernate de cache ObjectGrid, vous pouvez créer trois types de topologies : imbriquée, imbriquée et partitionnée, et distante.

Topologie imbriquée

Une topologie imbriquée crée un serveur eXtreme Scale dans l'espace de traitement de chaque application. Les plug-in OpenJPA et Hibernate lisent directement la copie en mémoire du cache et écrivent dans toutes les autres copies. Vous pouvez améliorer les performances d'écriture à l'aide de la réplication asynchrone. Cette topologie par défaut produit un résultat optimal lorsque la quantité de données mises en cache est suffisamment réduite pour être traitée par un seul processus.

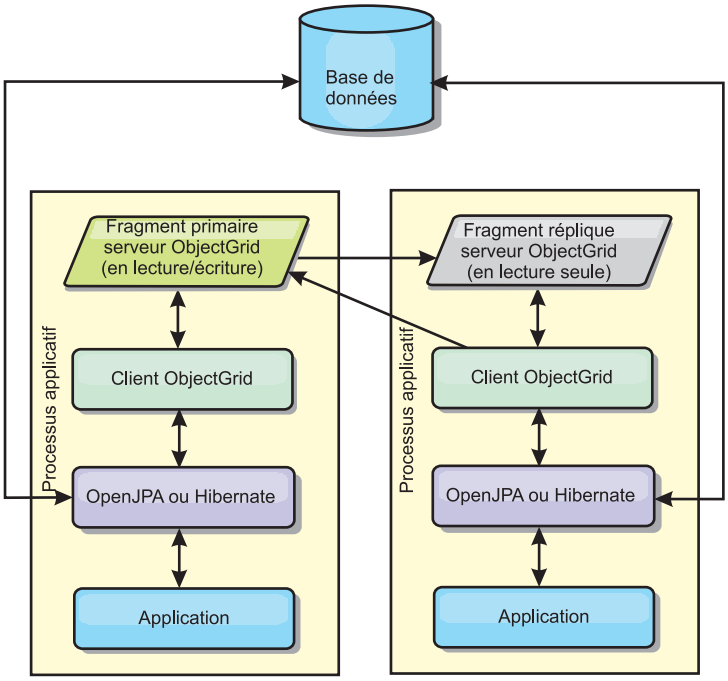


Figure 11. Topologie imbriquée JPA

Avantages :

- Toutes les lectures de cache sont très rapides, en accès local.
- Simple à configurer.

Limitations :

- La quantité de données est limitée à la taille du processus.
- Toutes les mises à jour de cache sont envoyées à un processus.

Topologie imbriquée et partitionnée

Lorsque les données mises en cache sont trop volumineuses pour être comprises dans un seul processus, la topologie imbriquée et partitionnée utilise les partitions ObjectGrid pour diviser les données en plusieurs processus. Les performances obtenues ne sont pas aussi élevées que celles de la topologie imbriquée car la plupart des lectures de cache sont distantes. Toutefois, cette option est utile en cas de temps d'attente élevé de la base de données.

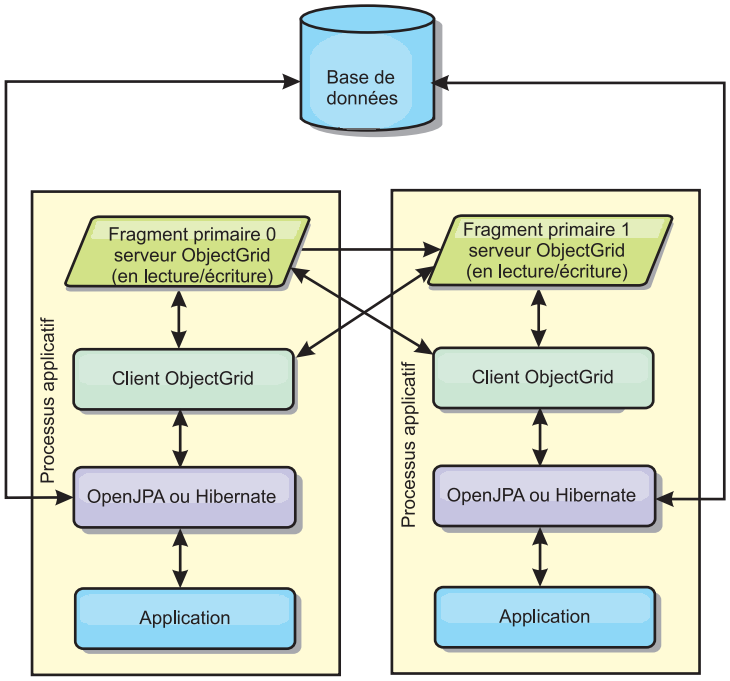


Figure 12. Topologie imbriquée et partitionnée JPA

Avantages :

- Stocke de grandes quantités de données.
- Simple à configurer.
- Les mises à jour de cache sont réparties sur plusieurs processus.

Limitation :

- La plupart des lectures et des mises à jour de cache sont distantes.

Par exemple, pour mettre en cache 10 Go de données avec 1 Go maximum par machine virtuelle Java, dix machines virtuelles Java sont requises. Le nombre de partitions doit par conséquent être défini sur 10 ou plus. De façon idéale, le nombre de partitions doit être défini sur un premier nombre dans lequel chaque fragment stocke une quantité de mémoire raisonnable. Le paramètre numberOfPartitions est généralement égal au nombre de machines virtuelles Java. Chaque machine virtuelle Java stocke une partition à l'aide de ce paramètre. Si vous activez la réplication, vous devez augmenter le nombre de machines virtuelles Java dans le système. Dans le cas contraire, chaque machine virtuelle Java stocke également une réplique de partition qui consomme autant de mémoire que la partition principale.

Consultez la rubrique relative à la définition de la taille de la mémoire et au calcul du nombre de partitions dans le *Guide d'administration* pour optimiser les performances de la configuration choisie.

Par exemple, dans un système comportant 4 machines virtuelles Java et dans lequel la valeur de paramètre numberOfPartitions est égale à 4, chaque machine virtuelle Java héberge une partition principale. Une opération de lecture a 25 pourcents de chances d'extraire des données d'une partition disponible en local, ce qui est sensiblement plus rapide qu'à partir d'une machine virtuelle Java distante. Si une opération de lecture, telle que l'exécution d'une requête, doit extraire une collection de données impliquant une répartition égale de quatre partitions, 75

pourcents des appels sont distants et 25 pourcents sont locaux. Si le paramètre ReplicaMode est défini sur `SYNC` ou `ASYNC` et si le paramètre ReplicaReadEnabled est défini sur `true`, quatre répliques de partitions sont créées et réparties entre quatre machines virtuelles Java. Chaque machine virtuelle Java héberge une partition principale et une réplique. L'opération de lecture a désormais à 50 pourcents de chances de s'exécuter en local. L'opération de lecture qui extrait une collection de données impliquant une répartition égale de quatre partitions comporte 50 pourcents d'appels distants et 50 pourcents d'appels locaux. Les appels locaux sont considérablement plus rapides que les appels distants. Dès que des appels distants sont effectués, les performances chutent.

Topologie distante

Une topologie distante stocke toutes les donnés mises en cache dans un ou plusieurs processus, ce qui réduit la sollicitation de la mémoire par les processus applicatifs. Vous pouvez tirer parti de la distribution de vos données sur des processus distincts en déployant une grille eXtreme Scale partitionnée et répliquée. Contrairement aux configurations imbriquée et imbriquée et partitionnée décrites précédemment, si vous voulez gérer la grille distante, vous devez l'effectuer indépendamment de l'application et du fournisseur JPA. Pour plus d'informations sur la gestion d'un déploiement de grille eXtreme Scale, consultez la rubrique relative à la surveillance de votre environnement de déploiement.

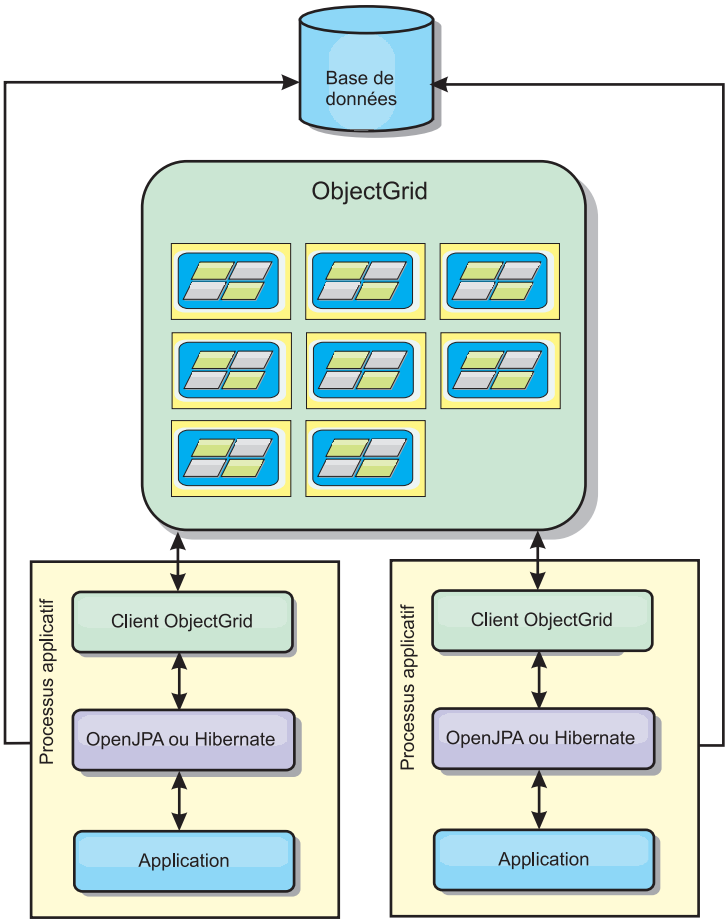


Figure 13. Topologie distante JPA

Avantages :

- Stocke de grandes quantités de données.
- Le processus applicatif est exempt de données en cache.
- Les mises à jour de cache sont réparties sur plusieurs processus.
- Options de configuration extrêmement souples.

Limitation :

- Toutes lectures et mises à jour de cache sont distantes.

Configuration du plug-in de cache Hibernate

Vous pouvez activer un cache eXtreme Scale pour Hibernate en définissant des propriétés dans le fichier de configuration.

7.1+ Pour l'intégration à WebSphere Application Server, le plug-in de cache Hibernate est packagé dans le fichier `oghibernate-cache.jar` et il est installé dans `WAS_HOME/optionalLibraries/ObjectGrid`. Pour pouvoir utiliser ce plug-in, vous devez inclure le fichier `oghibernate-cache.jar` dans la bibliothèque Hibernate. Ainsi, si vous incluez cette bibliothèque dans votre application, vous devez inclure également le fichier `oghibernate-cache.jar`. Si vous définissez une bibliothèque partagée où inclure la bibliothèque Hibernate, vous devrez placer le fichier `oghibernate-cache.jar` dans le répertoire de cette bibliothèque partagée.

7.1+ eXtreme Scale version 7.1 n'installe pas le fichier cglib.jar dans l'environnement WebSphere Application Server. Si des applications ou des bibliothèques partagées comme Hibernate dépendent de `cglib.jar`, repérez ce dernier et incluez-le dans le chemin d'accès aux classes. Si, par exemple, votre application inclut tous les fichiers JAR de la bibliothèque Hibernate à l'exclusion du `cglib.jar` fourni avec Hibernate, vous devrez inclure dans votre application le fichier `cglib.jar` d'Hibernate.

Paramètres

La syntaxe pour la définition de la propriété du fichier `persistence.xml` est la suivante :

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 syntaxe pour la définition de la propriété du fichier `hibernate.cfg.xml` est la suivante :

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 propriété provider_class correspond à la propriété com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider. Pour activer le cache de la requête, définissez la valeur sur `true` pour la propriété use_query_cache. Utilisez la propriété objectgrid.configuration pour spécifier les propriétés de configuration du cache d'eXtreme Scale.

Vous devez spécifier une valeur de propriété ObjectGridName unique pour éviter les conflits de nom. Les autres propriétés de configuration du cache d'eXtreme Scale sont facultatives.

La propriété objectgrid.hibernate.regionNames est facultative et doit être spécifiée lorsque les valeurs regionNames sont définies après l'initialisation du cache d'eXtreme Scale. Prenons l'exemple d'une classe d'entité mappée sur une valeur regionName dont la classe d'entité n'est pas spécifiée dans le fichier `persistence.xml` ou n'est pas incluse dans le fichier de mappage Hibernate. En outre, cette classe d'entité comporte une annotation d'entité. La valeur regionName pour cette classe d'entité est ainsi résolue au moment du chargement de la classe lors de l'initialisation du cache d'eXtreme Scale. La méthode Query.setCacheRegion(String regionName) exécutée après l'initialisation du cache d'eXtreme Scale illustre également cette configuration. Dans ces situations, incluez toutes les éventuelles valeurs regionNames déterminées de façon dynamique dans la propriété objectgrid.hibernate.regionNames de sorte que le cache d'eXtreme Scale puisse préparer les mappes de sauvegarde pour toutes les valeurs regionNames.

Voici des exemples de fichiers `persistence.xml` et `hibernate.cfg.xml` :

persistence.xml

```
<persistence-unit name="testPU2">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <class>com.ibm.websphere.objectgrid.jpa.test.Department</class>
    <properties>
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.connection.url" value="jdbc:derby:DB_testPU2;create=true" />
      <property name="hibernate.connection.driver_class" value="org.apache.derby.jdbc.EmbeddedDriver" />

      <property name="hibernate.cache.provider_class" value="com.ibm.websphere.objectgrid.
               hibernate.cache.ObjectGridHibernateCacheProvider" />
      <property name="hibernate.cache.use_query_cache" value="true"/>
      <property name="objectgrid.configuration" value="ObjectGridName=myOGName,ObjectGridType=
               EMBEDDED,MaxNumberOfReplicas=4" writeBehind=true, writeBehindInterval=5000,
               writeBehindPoolSize=10, writeBehindMaxBatchSize=1000" />
      <property name="objectgrid.hibernate.regionNames" value="queryRegion1, queryRegion2" />
    </properties>
</persistence-unit>
```

7.1+ La version 7.1 introduit des possibilités supplémentaires de configuration portant spécifiquement sur la fonction d'écriture différée pour le plug-in de cache Hibernate, qui viennent s'ajouter à celles du plug-in de cache JPA standard.

writeBehind

Valeurs valides : TRUE ou FALSE.

Valeur par défaut : FALSE.

Lorsque writeBehind est activé, les mises à jour sont provisoirement stockées dans un espace de stockage des données de portée JVM jusqu'à ce que soit remplie la condition writeBehindInterval ou la condition writeBehindMaxBatchSize.

Avertissement : Les autres paramètres de configuration de l'écriture différée sont ignorés sauf si writeBehind est activé.

writeBehindInterval

Valeurs valides : supérieur ou égal à 1.

Valeur par défaut : 5000 (5 secondes).

Spécifie en millisecondes l'intervalle séparant deux écritures de mises à jour vers le cache.

writeBehindPoolSize

Valeurs valides : supérieur ou égal à 1.

Valeur par défaut : 5.

Spécifie la taille maximale du pool d'unités d'exécution utilisé pour l'écriture des mises à jour vers le cache.

writeBehindMaxBatchSize

Valeurs valides : supérieur ou égal à 1.

Valeur par défaut : 1000.

Spécifie la taille maximale de lots par cache de région pour l'écriture des mises à jour vers le cache.

Notre exemple de code affichait la configuration suivante de la fonction write behind d'écriture différée :

```
writeBehind=true, writeBehindInterval=5000, writeBehindPoolSize=10, writeBehindMaxBatchSize=1000
```

où

- writeBehind=TRUE active la fonction d'écriture différée
- writeBehindInterval=5000 indique que les mises à jour doivent être écrites dans le cache environ toutes les 5 secondes
- writeBehindPoolSize=10 indique que le nombre maximal d'unités d'exécution utilisées est de 10 pour l'écriture des mises à jour vers le cache
- writeBehindMaxBatchSize=1000 indique que, si les mises à jour stockées dans le stockage d'écriture différée d'un cache de région dépasse le nombre de 1000 entrées, les mises à jour seront écrites vers le cache, même si l'intervalle spécifié pour la condition writeBehindInterval ne s'est pas écoulé. Autrement dit, les mises à jour seront écrites dans le cache soit toutes les 5 secondes environ, soit lorsque la taille du stockage d'écriture différée dans chacun des caches de région dépasse les 1000 entrées. Il convient de noter que, dans le cas où la condition writeBehindMaxBatchSize est remplie, seul le cache de région qui remplit cette condition écrira vers le cache ses mises à jour différées. Un cache de région correspond habituellement à une entité ou à une requête.

Important :

La configuration de la fonction d'écriture différée est à manier avec prudence. Elle rallonge en effet les temps d'attente de la synchronisation de données sur toutes les machines virtuelles Java et elle augmente le risque de pertes de mises à jour. Sur un système utilisant cette configuration avec au moins quatre machines virtuelles Java, la mise à jour effectuée sur l'une de ces quatre machines ne sera disponible pour les autres machines Java qu'au bout d'une quinzaine de secondes. Si deux des machines virtuelles Java actualisent la même entrée, la première à écrire la mise à jour dans le cache perdra cette mise à jour.

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>
```

Préchargement des données dans le cache d'ObjectGrid

Vous pouvez utiliser la méthode preload de la classeObjectGridHibernateCacheProvider pour précharger les données dans le cache d'ObjectGrid pour une classe d'entité.

Exemple 1

Utilisation d'EntityManagerFactory

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("testPU");
ObjectGridHibernateCacheProvider.preload("objectGridName", emf, TargetEntity.class, 100, 100);
```

Exemple 2

Utilisation de SessionFactory

```
org.hibernate.cfg.Configuration cfg = new Configuration();
// utilisez la méthode de configuration addResource, addClass et setProperty pour préparer
// la configuration requise pour créer SessionFactor
SessionFactory sessionFactory= cfg.buildSessionFactory();
ObjectGridHibernateCacheProvider.preload("objectGridName", sessionFactory,
TargetEntity.class, 100, 100);
```

Remarque :

1. Dans un système réparti, ce mécanisme de préchargement peut uniquement être appelé à partir d'une machine virtuelle Java. Le mécanisme de préchargement ne peut pas être exécuté à partir de plusieurs machines virtuelles Java.
2. Avant d'exécuter le préchargement, vous devez initialiser le cache d'eXtreme Scale en créant EntityManager à l'aide d'EntityManagerFactory afin d'obtenir toutes les mappes de sauvegarde correspondantes ; dans le cas contraire, le préchargement force l'initialisation du cache avec une seule mappe de sauvegarde par défaut pour prendre en charge toutes les entités. Une seule mappe de sauvegarde est ainsi partagée par toutes les entités.

Personnalisation de la configuration du cache Hibernate avec XML

Dans la plupart des cas, la définition des propriétés du cache est amplement suffisante. Toutefois, si vous souhaitez peaufiner la personnalisation de la grille d'objets utilisée par le cache, vous pouvez mettre à disposition dans votre répertoire `META-INF` des fichiers XML de configuration de Hibernate ObjectGrid, à la manière du fichier `persistence.xml`. Pendant l'initialisation, le cache recherche ces fichiers XML et les traite s'il les trouve.

Il existe trois types de fichiers XML de configuration Hibernate ObjectGrid : `hibernate-objectGrid.xml` (configuration de la grille d'objets, `hibernate-objectGridDeployment.xml` (règle de déploiement) et `hibernate-objectGrid-client-override.xml` (configuration des remplacements par les clients ObjectGrid). Pour personnaliser cette topologie, vous pouvez fournir le fichier XML adapté au type de l'eXtreme Scale configuré.

Pour le type EMBEDDED comme pour le type EMBEDDED_PARTITION, vous pouvez fournir n'importe lequel de ces trois fichiers XML pour personnaliser la grille d'objets, la règle de déploiement et la configuration des remplacements par les clients ObjectGrid.

Dans le cas d'un ObjectGrid REMOTE, le cache ne crée pas d'ObjectGrid dynamique. Le cache ne contient en fait qu'un ObjectGrid côté client provenant du service de catalogue. Dans ce cas, vous ne pouvez fournir qu'un fichier `hibernate-objectGrid-client-override.xml` qui personnalisera la configuration de la substitution de l'ObjectGrid client.

1. **Configuration de l'ObjectGrid** : utilisez le fichier `META-INF/hibernate-objectGrid.xml`. Ce fichier sert à personnaliser une configuration d'ObjectGrid de type EMBEDDED ou de type EMBEDDED_PARTITION. Si l'ObjectGrid est de type REMOTE, ce fichier est ignoré. Par défaut, chaque classe d'entité est associée à une valeur regionName (par défaut au nom de la classe d'entité) mappée sur une configuration BackingMap désignée sous regionName au sein de la configuration de l'ObjectGrid. Par exemple, la classe d'entité com.mycompany.Employee est associée à une valeur regionName qui a la valeur par défaut com.mycompany.Employee BackingMap. La configuration BackingMap par défaut est readOnly="false", copyKey="false", lockStrategy="NONE" et `copyMode="NO_COPY"`. Vous pouvez tout à fait personnaliser des mappes de sauvegarde avec la configuration que vous choisissez. Le mot clé réservé ALL_ENTITY_MAPS représente tous les mappages à l'exclusion des mappages personnalisés répertoriés dans le fichier `hibernate-objectGrid.xml`. Les mappes de sauvegarde qui ne figurent pas dans ce fichier `hibernate-objectGrid.xml` utilisent la configuration par défaut.
2. **Configuration de l'ObjectGridDeployment** : utilisez le fichier `META-INF/hibernate-objectGridDeployment.xml`. Ce fichier sert à personnaliser la règle de déploiement. Lorsque celle-ci est personnalisée, si le fichier `hibernate-objectGridDeployment.xml` est fourni, la règle de déploiement par défaut est ignorée. Toutes les valeurs d'attribut de la règle de déploiement proviennent du fichier `hibernate-objectGridDeployment.xml` fourni.
3. **Configuration des remplacements ObjectGrid par les clients :** utilisez le fichier `META-INF/hibernate-objectGrid-client-override.xml`. Ce fichier sert à personnaliser un ObjectGrid côté client. Par défaut, le cache de l'ObjectGrid applique une configuration par défaut de remplacement par les clients, qui désactive le cache local. Si l'application a besoin d'un cache local, elle peut fournir ce fichier en y spécifiant `numberOfBuckets="xxx"`. La substitution par défaut de clients désactive le cache local en définissant `numberOfBuckets="0"`. Pour activer le cache local, il suffit de donner à numberOfBuckets une valeur supérieure à 0 dans le fichier `hibernate-objectGrid-client-override.xml`. Le fonctionnement du fichier `hibernate-objectGrid-client-override.xml` est semblable à celui du fichier `hibernate-objectGrid.xml` : Il substitue ou étend la configuration par défaut des remplacements ObjectGrid par les clients.

Exemples de fichiers XML Hibernate ObjectGrid

Les fichiers XML Hibernate ObjectGrid doivent être créés à partir de la configuration de l'unité de persistance.

Voici à titre d'exemple un fichier `persistence.xml` représentant la configuration d'une unité de persistance :

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>
```

Voici le fichier `hibernate-objectGrid.xml` qui correspond au fichier `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>
```

Remarque : Les mappes org.hibernate.cache.UpdateTimestampsCache, org.hibernate.cache.StandardQueryCache et defaultCacheMap sont requises.

Voici le fichier `hibernate-objectGridDeployment.xml` qui correspond au fichier `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>
```

Remarque : Les mappes org.hibernate.cache.UpdateTimestampsCache, org.hibernate.cache.StandardQueryCache et defaultCacheMap sont requises.

Système externe de cache pour le type REMOTE d'ObjectGrid

Afin de pouvoir configurer un cache d'ObjectGrid de type REMOTE, vous devez configurer un système externe eXtreme Scale. Pour configurer ce système externe, vous aurez besoin des deux fichiers XML de configuration ObjectGrid et ObjectGridDeployment basés sur un fichier `persistence.xml`. Les fichiers XML de configuration Hibernate ObjectGrid et ObjectGridDeployment décrits dans la section relative aux exemples de fichiers Hibernate ObjectGrid sont tout à fait utilisables pour configurer un système externe eXtreme Scale.

Un tel système externe comporte le service de catalogue et les processus de serveurs ObjectGrid. Vous devez démarrer le service de catalogue avant les serveurs de conteneur. Pour plus d'informations, consultez les détails sur le démarrage des serveurs autonomes eXtreme Scale et les processus de conteneur dans le *Guide d'administration*.

Résolution des incidents

1. **`CacheException: Failed to get ObjectGrid server`**

   Avec un ObjectGrid de type EMBEDDED ou EMBEDDED_PARTITION, le cache tente d'obtenir une instance de serveur à partir de l'exécution eXtreme Scale. Dans un environnement Java Platform, Standard Edition, un serveur eXtreme Scale avec un service de catalogue intégré est démarré. Ce service de

catalogue essaie d'ausculter le port 2809. Cette erreur se produit si ce port est utilisé par un autre processus. Si des points de contact externes sont spécifiés pour le service de catalogue (avec le fichier `objectGridServer.properties`, par exemple), cette erreur se produit si le nom d'hôte ou le port sont spécifiés de manière erronée.

2. **CacheException : Failed to get REMOTE ObjectGrid for configured REMOTE ObjectGrid. objectGridName = [ObjectGridName], PU name = [persistenceUnitName]**

   Cette erreur se produit lorsque le cache échoue à obtenir un ObjectGrid à partir des points de contact du service de catalogue. L'erreur est causée en général par un nom d'hôte ou un port incorrects.

3. **CacheException: Cannot have two PUs [persistenceUnitName_1, persistenceUnitName_2] configured with same ObjectGridName [ObjectGridName] of EMBEDDED ObjectGridType**

   Cette exception résulte de la présence d'un grand nombre d'unités de persistance configurées alors que les caches de ces unités sont configurées avec le même nom d'ObjectGrid et un type EMBEDDED. Ces configurations d'unités de persistance peuvent être dans les mêmes fichiers `persistence.xml` ou dans des fichiers différents. Vous devez vérifier que le nom d'ObjectGrid est unique pour chacune des unités de persistance lorsque le type d'ObjectGridType est EMBEDDED.

4. **CacheException: REMOTE ObjectGrid [ObjectGridName] does not include required BackingMaps [mapName_1, mapName_2,...]**

   Avec un ObjectGrid de type REMOTE, cette exception se produit si l'ObjectGrid obtenu côté client ne dispose pas des BackingMaps d'entités complètes pour prendre en charge le cache pour l'unité de persistance. Supposons par exemple que cinq classes d'entités sont répertoriées dans la configuration des unités de persistance mais que l'ObjectGrid obtenu ne dispose que de deux BackingMaps. Même si l'ObjectGrid obtenu disposait de dix BackingMaps, il suffit que l'un des cinq BackingMaps d'entités requis soit introuvable dans ces dix BackingMaps pour que l'exception continue de se produire.

Configuration du plug-in de cache OpenJPA

WebSphere eXtreme Scale fournit les deux implémentations de cache pour OpenJPA : DataCache et QueryCache. Par cache OpenJPA ObjectGrid (ou ObjectGrid en abrégé), l'on désigne aussi bien l'implémentation DataCache que l'implémentation QueryCache.

Paramètres

L'activation ou la désactivation du cache d'eXtreme Scale pour OpenJPA s'effectue en définissant les propriétés de configuration openjpa.DataCache et openjpa.QueryCache dans le fichier `persistence.xml`. La syntaxe est la suivante :

```
<property name="openjpa.DataCache"
          value="<object_grid_datacache_class(<propriété>=<valeur>,...)"/>
<property name="openjpa.QueryCache"
          value="<object_grid_querycache_class(<propriété>=<valeur>,...)"/>
```

Les deux caches peuvent accepter des propriétés eXtreme Scale pour la configuration du cache utilisé par l'unité de persistance.

En plus du paramétrage du cache dans eXtreme Scale, la propriété openjpa.RemoteCommitProvider doit être définie avec `sjvm` comme valeur :

```
<property name="openjpa.RemoteCommitProvider" value="sjvm"/>
```

La valeur de délai d'expiration spécifiée avec l'annotation @DataCache pour chacune des classes d'entités est ramenée à celle de la mappe de sauvegarde vers laquelle chaque entité est mise en cache. Mais la valeur name spécifiée avec l'annotation @DataCache est ignorée par le cache d'eXtreme Scale. Le nom qualifié complet de la classe d'entité est le nom du mappage du cache.

Les méthodes pin et unpin d'OpenJPA StoreCache et QueryCache ne sont pas prises en charge et elle ne donnent rien.

Dans la liste de la classe de cache ObjectGrid, vous pouvez spécifier la propriété ObjectGridName, la propriété ObjectGridType ou toute autre propriétés en rapport avec des règles de déploiement simple pour personnaliser la configuration du cache. Voici un exemple :

```
<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"/>
```

Les configurations de DataCache et de QueryCache sont indépendantes l'une de l'autre. Vous pouvez activer l'une ou l'autre. Mais, si les deux configurations sont activées, la configuration de QueryCache utilisera celle de DataCache et ses propriétés à elle seront ignorées.

Personnaliser avec XML la configuration du cache OpenJPA

Dans la plupart des cas, la définition par eXtreme Scale des propriétés du cache est amplement suffisante. Mais, si vous souhaitez peaufiner la personnalisation de l'ObjectGrid utilisé par le cache, vous pouvez mettre à disposition dans votre répertoire META-INF des fichiers XML de configuration de l'OpenJPA ObjectGrid, à la manière du fichier `persistence.xml`. Pendant l'initialisation du cache, le cache ObjectGrid recherche ces fichiers XML et les traite s'il les trouve.

Il existe trois types de fichiers XML de configuration OpenJPA ObjectGrid : `openjpa-objectGrid.xml` (configuration de l'ObjectGrid), `openjpa-objectGridDeployment.xml` (règle de déploiement) et `openjpa-objectGrid-client-override.xml` (configuration de la substitution des ObjectGrid clients). Pour personnaliser l'ObjectGrid, vous fournirez le fichier XML approprié au type de l'ObjectGrid à configurer.

Pour le type EMBEDDED comme pour le type EMBEDDED_PARTITION, vous pouvez fournir n'importe lequel de ces trois fichiers XML pour personnaliser l'ObjectGrid, la règle de déploiement et la configuration de la substitution des ObjectGrid clients.

Dans le cas d'un ObjectGrid REMOTE, le cache ne crée pas d'ObjectGrid dynamique. Le cache ne contient en fait qu'un ObjectGrid côté client provenant du service de catalogue. Dans ce cas, vous pouvez fournir que le fichier `openjpa-objectGrid-client-override.xml` qui personnalisera la configuration de la substitution de l'ObjectGrid client.

1. **Configuration d'ObjectGrid :** utiliser le fichier `META-INF/openjpa-objectGrid.xml`. Ce fichier sert à personnaliser une configuration d'ObjectGrid de type EMBEDDED ou de type EMBEDDED_PARTITION. Si l'ObjectGrid est de type REMOTE, ce fichier est ignoré. Par défaut, chaque classe d'entité est mappée à sa propre configuration BackingMap désignée au sein de la

configuration de l'ObjectGrid sous le nom de la classe. Ainsi, la classe d'entité com.mycompany.Employee sera mappée à la configuration BackingMap com.mycompany.Employee. La configuration BackingMap par défaut est `readOnly="false"`, `copyKey="false"`, `lockStrategy="NONE"` et `copyMode="NO_COPY"`. Vous pouvez tout à fait personnaliser des mappes de sauvegarde avec la configuration que vous choisissez. Le mot clé réservé `ALL_ENTITY_MAPS` représente tous les mappages à l'exclusion des mappages personnalisés répertoriés dans le fichier `openjpa-objectGrid.xml`. Les mappes de sauvegarde qui ne figurent pas dans ce fichier utilisent la configuration par défaut. Si les mappes de sauvegarde personnalisées ne spécifient pas l'attribut ou les propriétés BackingMaps et que ces attributs sont spécifiés dans la configuration par défaut, ce sont les valeurs des attributs dans cette configuration qui s'appliquent. Par exemple, si une classe d'entité est annotée avec `timeToLive=30`, la configuration BackingMap par défaut de cette entité aura un `timeToLive=30`. Si le fichier personnalisé `openjpa-objectGrid.xml` inclut également la mappe de sauvegarde mais sans spécifier de valeur pour timeToLive value, la mappe personnalisée aura la valeur `timeToLive=30` qui est la valeur par défaut. Le fichier `openjpa-objectGrid.xml` a pour finalité de remplacer ou d'étendre la configuration par défaut.

2. **Configuration d'ObjectGridDeployment :** utiliser le fichier `META-INF/openjpa-objectGridDeployment.xml`. Ce fichier sert à personnaliser la règle de déploiement. Lorsque celle-ci est personnalisée, si le fichier `openjpa-objectGridDeployment.xml` est fourni, la règle de déploiement par défaut est ignorée. Toutes les valeurs des attributs de la règle de déploiement sont tirées du fichier `openjpa-objectGridDeployment.xml`.
3. **Configuration du remplacement des ObjectGrid par les clients :** utiliser le fichier `META-INF/openjpa-objectGrid-client-override.xml`. Ce fichier sert à personnaliser un ObjectGrid côté client. Par défaut, le cache de l'ObjectGrid applique une configuration par défaut de substitution des ObjectGrid par les clients, qui désactive les caches locaux (near cache). Si une application a besoin d'un cache local, elle peut fournir ce fichier en y spécifiant `numberOfBuckets="xxx"`. La substitution par défaut par les clients désactive le cache local en définissant `numberOfBuckets="0"`. Pour activer le cache local, il suffit de donner à numberOfBuckets une valeur supérieure à 0 dans le fichier `openjpa-objectGrid-client-override.xml`. Le fonctionnement du fichier `openjpa-objectGrid-client-override.xml` est semblable à celui du fichier `openjpa-objectGrid.xml`. Le fichier `openjpa-objectGrid-client-override.xml` remplace ou étend la configuration d'ObjectGrid.

Exemples de fichiers XML OpenJPA ObjectGrid

Les fichiers XML OpenJPA ObjectGrid XML doivent être créés à partir de la configuration de l'unité de persistance.

Voici à titre d'exemple un fichier `persistence.xml` représentant la configuration d'une unité de persistance :

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>
```

Et voici le fichier `openjpa-objectGrid.xml` correspondant à ce fichier `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>
```

Remarque :

1. Chaque entité est mappée à une mappe de sauvegarde qui porte le nom qualifié complet de la classe de cette entité.
2. Si les classes d'entités sont dans une hiérarchie d'héritage, les classes enfants se mappent à la mappe de sauvegarde parent. La hiérarchie d'héritage partage une même mappe de sauvegarde.
3. Le mappage ObjectGridQueryCache map est indispensable pour la prise en charge de QueryCache.
4. L'ObjectTransformer de la backingMapPluginCollection de chaque mappage d'entrée doit utiliser la classe com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer.
5. L'index de clé de la backingMapPluginCollection d'un mappage d'ObjectGridQueryCache être nommé QueryCacheKeyIndex (voir l'exemple).
6. L'expulseur (evictor) est facultatif pou chaque mappage.

Voici le fichier `openjpa-objectGridDeployment.xml` qui correspond au fichier `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>
```

Remarque : Le mappage ObjectGridQueryCache map est indispensable pour la prise en charge de QueryCache.

Système externe de cache pour le type REMOTE d'ObjectGrid

Afin de pouvoir configurer un cache d'ObjectGrid de type REMOTE, il vous faut configurer un système externe. Pour configurer ce système externe, vous aurez besoin des deux fichiers XML de configuration ObjectGrid et ObjectGridDeployment basés sur un fichier `persistence.xml`. Les exemples de fichiers OpenJPA ObjectGrid et ObjectGridDeployment décrits plus haut sont tout à fait utilisables pour configurer un système externe eXtreme Scale.

Un tel système externe comporte aussi bien le service de catalogue que des processus de serveurs conteneurs. Vous devez démarrer le serveur de catalogue avant les serveurs conteneurs.

Résolution des incidents

1. **CacheException: Failed to get ObjectGrid server**

   Avec un ObjectGrid de type EMBEDDED ou EMBEDDED_PARTITION, le cache d'eXtreme Scale essaie d'obtenir une instance du serveur à partir de l'exécution. Dans un environnement Java Platform, Standard Edition, un serveur eXtreme Scale avec un service de catalogue intégré est démarré. Ce service de catalogue essaie d'ausculter le port 2809. Cette erreur se produit si ce port est utilisé par un autre processus. Si des points de contact externes sont spécifiés pour le service de catalogue (avec le fichier `objectGridServer.properties`, par exemple), cette erreur se produit si le nom d'hôte ou le port sont spécifiés de manière erronée.

2. **CacheException: Failed to get REMOTE ObjectGrid for configured REMOTE ObjectGrid. objectGridName = [ObjectGridName], PU name = [persistenceUnitName]**

   Cette erreur se produit lorsque le cache échoue à obtenir un ObjectGrid à partir des points de contact du service de catalogue. L'erreur est causée en général par un nom d'hôte ou un port incorrects.

3. **CacheException: Cannot have two PUs [persistenceUnitName_1, persistenceUnitName_2] configured with same ObjectGridName [ObjectGridName] of EMBEDDED ObjectGridType**

   Cette exception résulte de la présence d'un grand nombre d'unités de persistance configurées alors que les caches eXtreme Scale de ces unités sont configurées avec le même nom d'ObjectGrid name et un type EMBEDDED. Ces configurations d'unités de persistance peuvent être dans les mêmes fichiers `persistence.xml` ou dans des fichiers différents. Vous devez vérifier que le nom d'ObjectGrid name est unique pour chacune des unités de persistance lorsque le type d'ObjectGridType est EMBEDDED.

4. **CacheException: REMOTE ObjectGrid [ObjectGridName] does not include required BackingMaps [mapName_1, mapName_2,...]**

   Avec un ObjectGrid de type REMOTE, cette exception se produit si l'ObjectGrid obtenu côté client ne dispose pas des mappes de sauvegarde complètes d'entités pour prendre en charge l'unité de persistance. Supposons par exemple que cinq classes d'entités sont répertoriées dans la configuration des unités de persistance mais que l'ObjectGrid obtenu ne dispose que de deux mappes de sauvegarde. Même si l'ObjectGrid obtenu disposait de dix mappes de sauvegarde, il suffit que l'une des cinq mappes d'entités requises soit introuvable dans ces dix mappes pour que l'exception continue de se produire.

Remarque : Le format des données du cache OpenJPA d'eXtreme Scale a été modifié dans un souci d'optimiser ses performances. Tous les systèmes qui hébergent des applications OpenJPA configurées avec eXtreme Scale comme cache de niveau 2 doivent être arrêtés avant le passage à la version 7.0 de WebSphere eXtreme Scale.

Configuration de gestionnaires de sessions HTTP

Le gestionnaire de sessions HTTP offre des fonctions de réplication de sessions pour une application associée. Le gestionnaire de réplication de session collabore avec le conteneur Web pour créer des sessions HTTP et gérer les cycles de vie des sessions HTTP qui sont associées à l'application.

Fichiers XML de configuration du gestionnaire de sessions HTTP

Lorsqu'on démarre un serveur conteneur qui stocke des données de session HTTP, l'on peut soit utiliser les fichiers XML par défaut, soit spécifier des fichiers XML personnalisés qui créent des noms ObjectGrid spécifiques, des nombres de répliques particularisés, etc.

Emplacement des fichiers d'exemples

Ces fichiers XML sont packagés dans *<extremescale>*/ObjectGrid/session/samples pour les installations autonomes ou dans *<racine_install_WAS>*/optionalLibraries/ObjectGrid/session/samples pour les installations de WebSphere eXtreme Scale dans une cellule WebSphere Application Server.

Package XML imbriqué

Dans un scénario imbriqué, dans lequel le serveur conteneur démarre dans le groupe des serveurs Web conteneurs, les fichiers objectGrid.xml et objectGridDeployment.xml sont founis par défaut. Vous pouvez tout à fait modifier ces fichiers pour personnaliser le comportement du gestionnaire de sessions HTTP.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd" xmlns="http://ibm.com/ws/objectgrid/config">
 <objectGrids>
  <objectGrid name="session">
   <bean id="ObjectGridEventListener" className="com.ibm.ws.xs.sessionmanager.SessionHandleManager"/>
   <backingMap name="objectgridSessionMetadata" pluginCollectionRef="objectgridSessionMetadata" readOnly="false"
     lockStrategy="PESSIMISTIC" ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3600" copyMode="NO_COPY"/>
   <backingMap name="objectgridSessionAttribute.*" template="true" readOnly="false" lockStrategy="PESSIMISTIC"
     ttlEvictorType="NONE" copyMode="NO_COPY"/>
       <backingMap name="objectgridSessionTTL.*" template="true" readOnly="false" lockStrategy="PESSIMISTIC"
     ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3600" copyMode="NO_COPY"/>
  </objectGrid>
 </objectGrids>
    <backingMapPluginCollections>
        <backingMapPluginCollection id="objectgridSessionMetadata">
            <bean id="MapEventListener" className="com.ibm.ws.xs.sessionmanager.MetadataMapListener"/>
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

Valeurs modifiables :

- **Attribut name de l'ObjectGrid :** doit être identique à la propriété objectGridName dans le fichier `splicer.properties` qui sert à faire un raccord entre l'application Web et l'attribut objectgridName dans le fichier `objectGridDeployment.xml`. Si vous avez plusieurs applications et que vous voulez stocker les données de session dans différentes grilles, ces applications doivent avoir différentes valeurs pour l'attribut name de l'ObjectGrid. Le nom de grille est la seule chose qui peut être modifiée dans ce fichier.

Valeurs non modifiables :

- **ObjectGridEventListener :** la ligne ObjectGridEventListener ne peut être modifiée et elle est utilisée en interne.
- **objectgridSessionMetadata :** la ligne objectgridSessionMetadata se réfère à la mappe dans laquelle sont stockées les métadonnées des sessions HTTP. Il y a une entrée pour chaque session HTTP stockée dans la grille dans cette mappe.
- **objectgridSessionAttribute.* :** la ligne objectgridSessionAttribute.* définit une mappe dynamique qui sert à créer la mappe dans laquelle sont stockés les attributs des sessions HTTP lorsque le paramètre **fragmentedSession** a la valeur `true` dans le fichier `splicer.properties` servant à raccorder l'application Web. Cette mappe dynamique est appelée objectgridSessionAttribute. Une autre mappe est créée à partir de ce modèle, qui est appelée objectgridSessionAttributeEvicted. Elle stocke les sessions qui ont expiré mais que le conteneur Web n'a pas invalidées.
- La ligne **MetadataMapListener** est à usage interne et elle ne peut être modifiée.

Figure 14. Fichier objectGrid.xml

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

 <objectgridDeployment objectgridName="session">
  <mapSet name="sessionMapSet" numberOfPartitions="5" minSyncReplicas="0" maxSyncReplicas="0"
    maxAsyncReplicas="1" developmentMode="false" placementStrategy="PER_CONTAINER">
             <map ref="objectgridSessionMetadata"/>
             <map ref="objectgridSessionAttribute.*"/>
             <map ref="objectgridSessionTTL.*"/>
    </mapSet>
   </objectgridDeployment>
</deploymentPolicy>
```

Valeurs modifiables :

- **Attribut objectgridName :** doit être identique à la propriété objectGridName property dans le fichier `splicer.properties` qui sert à raccorder l'application Web et l'attribut ObjectGrid dans le fichier `objectGrid.xml`.
- **Attributs de l'élément mapSet :** vous pouvez modifier toutes les propriétés mapSet à l'exception de l'attribut placementStrategy.

  Name Peut être modifié et prendre n'importe quelle valeur.

  numberOfPartitions
  Spécifie le nombre de partitions primaires qui sont démarrées sur chacun des serveurs hébergeant l'application Web. Au fur et à mesure que l'on ajoute des partitions, les données sont de plus en plus réparties dans l'éventualité d'un basculement. La valeur par défaut est de 5 partitions, ce qui convient parfaitement à la plupart des cas.

  minSyncReplicas, maxSyncReplicas et maxAsyncReplicas
  Spécifient le nombre et le type des fragments répliques qui stockent les données de session HTTP. La valeur par défaut est de 1 fragment réplique asynchrone, ce qui convient parfaitement à la plupart des cas. La réplication synchrone intervient pendant le chemin de demande, ce qui peut augmenter les temps de réponse de l'application Web.

  developmentMode
  Informe le service de placement eXtreme Scale si les fragments répliques d'une partition peuvent être positionnés sur le même noeud que leur fragment primaire. La valeur peut être définie comme true dans un environnement de développement, mais il est conseillé de désactiver cette fonction en environnement de production en raison des risques de pertes de données que pourrait provoquer une défaillance du noeud.

  placementStrategy
  Ne modifiez pas la valeur de cet attribut.
- Le reste du fichier se réfère aux mêmes noms de mappes que dans le fichier objectGrid.xml. Ces noms ne peuvent être modifiés.

Valeurs non modifiables :

- L'attribut placementStrategy dans l'élément mapSet.

Figure 15. Fichier objectGridDeployment.xml

Package XML distant

En mode distant, dans lequel les conteneurs s'exécutent en tant que processus autonomes, vous devez utiliser les fichiers `objectGridStandAlone.xml` et `objectGridDeploymentStandAlone.xml` pour démarrer les processus. Vous pouvez modifier ces fichiers pour adapter la configuration.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
 <objectGrids>
  <objectGrid name="session">
   <bean id="ObjectGridEventListener" className="com.ibm.ws.xs.sessionmanager.SessionHandleManager"/>
           <backingMap name="objectgridSessionMetadata" pluginCollectionRef="objectgridSessionMetadata"
       readOnly="false" lockStrategy="PESSIMISTIC" ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3600"
       copyMode="COPY_TO_BYTES"/>
           <backingMap name="objectgridSessionAttribute.*" template="true" readOnly="false" lockStrategy="PESSIMISTIC"
       ttlEvictorType="NONE" copyMode="COPY_TO_BYTES"/>
           <backingMap name="objectgridSessionTTL.*" template="true" readOnly="false" lockStrategy="PESSIMISTIC"
       ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3600" copyMode="COPY_TO_BYTES"/>
  </objectGrid>
 </objectGrids>
    <backingMapPluginCollections>
        <backingMapPluginCollection id="objectgridSessionMetadata">
            <bean id="MapEventListener" className="com.ibm.ws.xs.sessionmanager.MetadataMapListener"/>
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

Valeurs modifiables :

- **Attribut objectgridName :** est modifiable, mais doit être identique à la propriété objectGridName property dans le fichier `splicer.properties` qui sert à raccorder l'application Web et l'attribut objectgridName dans le fichier `objectGridDeploymentStandAlone.xml`. Si vous avez plusieurs applications et que vous voulez stocker les données de session dans différentes grilles, ces applications doivent avoir des noms de grilles différents. Le nom de grille est la seule chose qui peut être modifiée dans ce fichier.

Valeurs non modifiables :

- **ObjectGridEventListener :** la ligne ObjectGridEventListener ne peut être modifiée et elle est utilisée en interne.
- **objectgridSessionMetadata :** la ligne objectgridSessionMetadata se réfère à la mappe dans laquelle sont stockées les métadonnées des sessions HTTP. Il y a une entrée pour chaque session HTTP stockée dans la grille dans cette mappe.
- **objectgridSessionAttribute.* :** la ligne objectgridSessionAttribute.* définit une mappe dynamique qui sert à créer la mappe dans laquelle sont stockés les attributs des sessions HTTP lorsque le paramètre **fragmentedSession** a la valeur `true` dans le fichier `splicer.properties` servant à raccorder l'application Web. Cette mappe dynamique est appelée objectgridSessionAttribute. Une autre mappe est créée à partir de ce modèle, qui est appelée objectgridSessionAttributeEvicted. Elle stocke les sessions qui ont expiré mais que le conteneur Web n'a pas invalidées.
- La ligne **MetadataMapListener** est à usage interne et elle ne peut être modifiée.

Figure 16. objectGridStandAlone.xml

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

 <objectgridDeployment objectgridName="session">
  <mapSet name="sessionMapSet" numberOfPartitions="5" minSyncReplicas="0" maxSyncReplicas="0"
   maxAsyncReplicas="1" developmentMode="false" placementStrategy="PER_CONTAINER">
            <map ref="objectgridSessionMetadata"/>
            <map ref="objectgridSessionAttribute.*"/>
            <map ref="objectgridSessionTTL.*"/>
     </mapSet>
    </objectgridDeployment>
</deploymentPolicy>
```

Valeurs modifiables :

- **Attribut objectgridName :** doit être identique à la propriété objectGridName property dans le fichier `splicer.properties` qui sert à raccorder l'application Web et l'attribut ObjectGrid dans le fichier `objectGrid.xml`.
- **Attributs de l'élément mapSet :** vous pouvez modifier toutes les propriétés mapSet à l'exception de l'attribut placementStrategy.

  Name Peut être modifié et prendre n'importe quelle valeur.

  numberOfPartitions
  Spécifie le nombre de partitions primaires qui sont démarrées sur chacun des serveurs hébergeant l'application Web. Au fur et à mesure que l'on ajoute des partitions, les données sont de plus en plus réparties dans l'éventualité d'un basculement. La valeur par défaut est de 5 partitions, ce qui convient parfaitement à la plupart des cas.

  minSyncReplicas, maxSyncReplicas et maxAsyncReplicas
  Spécifient le nombre et le type des fragments répliques qui stockent les données de session HTTP. La valeur par défaut est de 1 fragment réplique asynchrone, ce qui convient parfaitement à la plupart des cas. La réplication synchrone intervient pendant le chemin de demande, ce qui peut augmenter les temps de réponse de l'application Web.

  developmentMode
  Informe le service de placement eXtreme Scale si les fragments répliques d'une partition peuvent être positionnés sur le même noeud que leur fragment primaire. La valeur peut être définie comme true dans un environnement de développement, mais il est conseillé de désactiver cette fonction en environnement de production en raison des risques de pertes de données que pourrait provoquer une défaillance du noeud.

  placementStrategy
  Ne modifiez pas la valeur de cet attribut.
- Le reste du fichier se réfère aux mêmes noms de mappes que dans le fichier `objectGrid.xml`. Ces noms ne peuvent être modifiés.

Valeurs non modifiables :

- L'attribut placementStrategy dans l'élément mapSet.

Figure 17. objectGridDeploymentStandAlone.xml :

Configuration du gestionnaire de sessions HTTP avec WebSphere Application Server

WebSphere Application Server offre une fonction de gestion de session, mais cette dernière ne permet pas de traiter les charges de demandes extrêmes. WebSphere eXtreme Scale est livré avec une implémentation de la gestion des sessions qui fournit la réplication de sessions, la haute disponibilité, une meilleure évolutivité et des options de configuration plus robustes.

Avant de commencer

- WebSphere eXtreme Scale doit être installé dans votre cellule WebSphere Application Server ou WebSphere Application Server Network Deployment pour pouvoir utiliser le gestionnaire de sessions d'eXtreme Scale. Pour plus d'informations, voir «Intégration de WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client à WebSphere Application Server», à la page 22.

Pourquoi et quand exécuter cette tâche

Le gestionnaire de sessions HTTP de WebSphere eXtreme Scale prend en charge les serveurs imbriqués et éloignés pour la mise en cache.

Scénario de serveurs imbriqués

Dans ce scénario, les serveurs WebSphere eXtreme Scale sont regroupés dans les processus où les servlets sont exécutés. Le gestionnaire de sessions peut communiquer directement avec l'instance ObjectGrid locale, pour éviter les retards coûteux du réseau.

Si vous utilisez WebSphere Application Server, placez dans les répertoires `META-INF` de vos fichiers d'archive Web (WAR) les fichiers `objectgridRoot/session/samples/objectGrid.xml` et `objectgridRoot/session/samples/objectGridDeployment.xml` fournis. eXtreme Scale détecte automatiquement ces fichiers au démarrage de l'application et démarre automatiquement les conteneurs eXtreme Scale dans le même processus que le gestionnaire de sessions.

Vous pouvez modifier le fichier `objectGridDeployment.xml` suivant que vous souhaitiez utiliser une réplication synchrone ou asynchrone et en fonction du nombre de fragments répliques à configurer.

Scénario de serveurs éloignés

Dans les scénarios de serveurs distants, les serveurs conteneurs s'exécutent dans des processus différents que les servlets. Le gestionnaire de sessions communique avec un serveur conteneur distant. Pour pouvoir utiliser un serveur distant connecté à un réseau, le gestionnaire de sessions doit être configuré avec les noms d'hôte et les numéros de port du domaine de services de catalogue. Le gestionnaire de sessions utilise ensuite une connexion client eXtreme Scale pour communiquer avec le serveur de catalogues et avec les serveurs conteneurs.

Si les serveurs conteneurs doivent être démarrés dans des processus autonomes indépendants, démarrez les conteneurs eXtreme Scale à l'aide des fichiers `objectGridStandAlone.xml` et `objectGridDeploymentStandAlone.xml` fournis dans le répertoire des exemples du gestionnaire de sessions.

Procédure

1. Raccordez votre application de sorte qu'elle puisse utiliser le gestionnaire de sessions. Pour utiliser le gestionnaire de sessions, vous devez ajouter les déclarations de filtre appropriées aux descripteurs de déploiement Web de l'application. En outre, les paramètres de configuration du gestionnaire de sessions sont transmis au gestionnaire de sessions sous la forme de paramètres d'initialisation du contexte de servlet dans les descripteurs de déploiement. Ces informations peuvent être injectées dans votre application de bien des manières :

   - **7.1+ Sur la console d'administration de WebSphere Application Server**

Vous pouvez configurer votre application pour qu'elle utilise un gestionnaire WebSphere eXtreme Scale de sessions HTTP lorsque vous installez votre application. Vous pouvez également modifier la configuration de l'application ou du serveur pour qu'ils utilisent le gestionnaire WebSphere eXtreme Scale de sessions HTTP. Pour plus d'informations, voir «Création de la persistance des sessions dans une grille de données», à la page 264.

Remarque : Cette possibilité ne fonctionne que si tous les noeuds exécutant l'application contiennent le fichier `splicer.properties` dans le même chemin. Dans le cas d'environnements mixtes contenant des noeuds Windows et UNIX, cette manière de procéder n'est pas possible et vous devez raccorder manuellement l'application.

- **Option de raccordement automatique**

  Vous n'avez pas besoin de raccorder manuellement vos applications lorsqu'elles s'exécutent dans WebSphere Application Server ou dans WebSphere Application Server Network Deployment. Vous pouvez ajouter une propriété personnalisée à une cellule ou un serveur, qui affectera toutes les applications Web de cette portée. Le nom de cette propriété est `com.ibm.websphere.xs.sessionFilterProps` et sa valeur doit correspondre à l'emplacement du fichier `splicer.properties` requis par vos applications.

  Voici un exemple de chemin d'accès pour l'emplacement d'un fichier : `/opt/splicer.properties`.

  – **Pour spécifier que toutes les applications Web d'une cellule sont raccordées, utilisez la console d'administration :**

    Allez à **Administration système** → **Cellule** → **Propriétés personnalisées** et ajoutez la propriété.

  – **Pour spécifier que toutes les applications Web d'un serveur d'applications donné sont raccordées, utilisez la console d'administration :**

    Allez à **Serveur d'applications** → ***<nom_serveur>*** → **Administration** → **Propriétés personnalisées** et ajoutez la propriété.

  Les portées cellule et serveur sont les seules portées utilisables et elles ne le sont que lors d'une exécution dans un gestionnaire de déploiement. Si vous avez besoin d'une autre portée, vous devez raccorder manuellement vos applications Web.

  Remarque : Le raccordement automatique ne fonctionne que si tous les noeuds exécutant l'application contiennent le fichier `splicer.properties` dans le même chemin. Dans le cas d'environnements mixtes contenant des noeuds Windows et UNIX, cette manière de procéder n'est pas possible et vous devez raccorder manuellement l'application.

- **Script addObjectGridFilter**

  Utilisez un script de ligne de commande fourni avec eXtreme Scale pour raccorder une application avec des déclarations de filtre et une configuration sous forme de paramètres d'initialisation du contexte de servlet. Ce script, `objectgridRoot/bin/addObjectGridFilter.sh` ou `objectgridRoot/session/bin/addObjectGridFilter.bat`, accepte deux paramètres : l'application (chemin d'accès absolu au fichier d'archive d'entreprise) qui doit faire l'objet d'une opération splice et le chemin d'accès absolu au fichier de propriétés qui contient diverses propriétés de configuration. La syntaxe de ce script est la suivante :

  Windows

  ```
  addObjectGridFilter.bat [location of ear file] [location of properties file]
  ```

UNIX

```
addObjectGridFilter.sh [location of ear file] [location of properties file]
```

UNIX **Exemple d'utilisation du produit eXtreme Scale installé sur WebSphere Application Server sous Unix :**

a. `cd objectgridRoot/optionalLibraries/ObjectGrid/session/bin`

b. `addObjectGridFilter.sh /tmp/mySessionTest.ear wasRoot/optionalLibraries/ObjectGrid/session/samples/splicer.properties`

UNIX **Exemple d'utilisation du produit eXtreme Scale installeé dans un répertoire autonome sous Unix :**

a. `cd objectgridRoot/session/bin`

b. `addObjectGridFilter.sh /tmp/mySessionTest.ear objectgridRoot/session/samples/splicer.properties`

Le filtre de servlet qui est joint conserve les valeurs de configuration par défaut. Vous pouvez remplacer ces valeurs par défaut par des options de configuration que vous spécifiez dans le fichier de propriétés, dans le second argument. Pour une liste des paramètres que vous pouvez utiliser, voir «Paramètres d'initialisation du contexte de servlet», à la page 271.

Vous pouvez modifier et utiliser l'exemple de fichier `splicer.properties` fourni avec l'installation d'eXtreme Scale. Vous pouvez également utiliser le script addObjectGridServlets, qui insère le gestionnaire de sessions en étendant chaque servlet. Mais le script recommandé est le script addObjectGridFilter.

- **Script de génération Ant**

  WebSphere eXtreme Scale est fourni avec un fichier `build.xml` qui peut être utilisé par Apache Ant, qui est inclus dans le dossier `wasRoot/bin` d'une installation WebSphere Application Server. Le fichier `build.xml` peut être modifié pour changer les propriétés de configuration du gestionnaire de sessions. Les propriétés de configuration sont identiques aux noms de propriété dans le fichier `splicer.properties`. Après que le fichier `build.xml` a été modifié, appelez le processus Ant en exécutant :

  – UNIX `ant.sh`, `ws_ant.sh`

  – Windows `ant.bat`, `ws_ant.bat`

  (UNIX) ou (Windows).

- **Mise à jour manuelle du descripteur Web**

  Editez le fichier `web.xml` intégré à l'application Web pour incorporer la déclaration de filtre, son mappage de servlet et les paramètres d'initialisation du contexte de servlet. Vous ne devez pas utiliser cette méthode car elle est source d'erreurs.

Pour une liste des paramètres que vous pouvez utiliser, voir «Paramètres d'initialisation du contexte de servlet», à la page 271.

2. Déployez l'application. Déployez l' application à l'aide de votre procédure normale pour un serveur ou un cluster. Une fois que vous avez déployé l'application, vous pouvez la démarrer.
3. Accédez à l'application. Vous pouvez maintenant accéder à l'application, qui interagit avec le gestionnaire de sessions et WebSphere eXtreme Scale.

Que faire ensuite

Vous pouvez modifier la majorité des attributs de configuration du gestionnaire de sessions lorsque vous instrumentez votre application pour utiliser le gestionnaire de sessions. Ces attributs sont : réplication synchrone ou asynchrone, longueur de l'ID session, taille de la table des sessions en mémoire, etc. En dehors des attributs modifiables lors de l'instrumentation de l'application, les seuls autres attributs de configuration que vous pouvez modifier après le déploiement de l'application sont ceux liés à la topologie des clusters de serveurs WebSphere eXtreme Scale et à la manière dont leurs clients (gestionnaires de sessions) s'y connectent.

Création de la persistance des sessions dans une grille de données :

Vous pouvez configurer votre application WebSphere Application Server pour stocker les sessions dans une grille de données. Cette grille de données peut se trouver dans un serveur de conteneur imbriqué exécuté dans WebSphere Application Server ou dans une grille de données éloignée.

Avant de commencer

Avant de modifier la configuration dans WebSphere Application Server, vous devez disposer des informations suivantes :

- Nom de la grille de données de session à utiliser. Pour plus d'informations sur la création d'une grille de données de session, voir «Configuration du gestionnaire de sessions HTTP avec WebSphere Application Server», à la page 260.
- Si le service de catalogue à utiliser pour gérer vos sessions se trouve en dehors de la cellule dans laquelle vous installez votre application de session, vous devez créer un domaine de service de catalogue. Pour plus d'informations, voir «Création de domaines de service de catalogue dans WebSphere Application Server», à la page 346.

Procédure

- **Pour configurer la gestion de session lors de l'installation de l'application, effectuez la procédure suivante :**
  1. Dans la console d'administration de WebSphere Application Server, cliquez sur **Applications** → **Nouvelle application** → **Nouvelle application d'entreprise**. Sélectionnez le chemin **Détaillé** pour la création de l'application, puis effectuez les premières étapes de l'assistant.
  2. Dans l'étape **Paramètres de gestion des sessions eXtreme Scale** de l'assistant, configurez la grille de données à utiliser. Choisissez la **Grille de données distante eXtreme Scale** ou la **Grille de données imbriquée eXtreme Scale**.
     - Pour l'option **Grille de données distante eXtreme Scale**, choisissez le domaine de service de catalogue qui gère la grille de données de session et choisissez une grille de données dans la liste des grilles de données de session.
     - Pour l'option **Grille de données imbriquée eXtreme Scale**, choisissez la configuration ObjectGrid par défaut ou spécifiez l'emplacement spécifique de vos fichiers de configuration ObjectGrid.
  3. Terminez l'installation de l'application en effectuant la procédure de l'assistant.

  Vous pouvez également installer l'application à l'aide d'un script wsadmin. Dans l'exemple suivant, le paramètre**-SessionManagement** crée une configuration identique à celle que vous pouvez créer dans la console d'administration :

Pour la configuration de la grille de données eXtreme Scale éloignée :
Pour le scénario imbriqué d'eXtreme Scale avec la configuration par défaut :

```
AdminApp.install('C:/A.ear',
'[  -nopreCompileJSPs -distributeApp
-nouseMetaDataFromBinary -nodeployejb -appname A -edition 8.0
-createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
off -noprocessEmbeddedConfig -filepermission .*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755
-buildVersion Unknown -noallowDispatchRemoteInclude -noallowServiceRemoteInclude
-asyncRequestDispatchType DISABLED -nouseAutoLink -SessionManagement [[true
XSRemoteSessionManagement cs0:!:grid0]]
-MapWebModToVH [[MicroWebApp microwebapp.war,WEB-INF/web.xml default_host] [MicroSipApp
microsipapp.war,WEB-INF/web.xml default_host] [MicroDG1App microdg1app.war,WEB-INF/web.xml
default_host] [MicroDG2App microdg2app.war,WEB-INF/web.xml default_host] [MicroSip2App
microsip2app.war,WEB-INF/web.xml default_host]]]')
```

Pour le scénario imbriqué d'eXtreme Scale avec une configuration

```
AdminApp.install('C:/A.ear', '[  -nopreCompileJSPs -distributeApp
-nouseMetaDataFromBinary -nodeployejb -appname A -edition 8.0
-createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
off -noprocessEmbeddedConfig -filepermission .*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755
-buildVersion Unknown -noallowDispatchRemoteInclude -noallowServiceRemoteInclude
-asyncRequestDispatchType DISABLED -nouseAutoLink -SessionManagement [[true
XSRemoteSessionManagement  :!: :!:default]]  -MapWebModToVH [[MicroWebApp microwebapp.war,WEB-INF/web.xml default_host]

[MicroSipApp
microsipapp.war,WEB-INF/web.xml default_host] [MicroDG1App microdg1app.war,WEB-INF/web.xml
default_host] [MicroDG2App microdg2app.war,WEB-INF/web.xml default_host] [MicroSip2App
microsip2app.war,WEB-INF/web.xml default_host]]]')
```

personnalisée :

```
AdminApp.install('C:/A.ear', '[  -nopreCompileJSPs -distributeApp
-nouseMetaDataFromBinary -nodeployejb -appname A -edition 8.0
-createMBeansForResources -noreloadEnabled -nodeployws -validateinstall
off -noprocessEmbeddedConfig -filepermission .*\.dll=755#.*\.so=755#.*\.a=755#.*\.sl=755
-buildVersion Unknown -noallowDispatchRemoteInclude -noallowServiceRemoteInclude
-asyncRequestDispatchType DISABLED -nouseAutoLink -SessionManagement [[true
XSRemoteSessionManagement  :!: :!:custom:!:c:\XS\objectgrid.xml:!:c:\XS\objectgriddeployment.xml]]
-MapWebModToVH [[MicroWebApp microwebapp.war,WEB-INF/web.xml default_host] [MicroSipApp
microsipapp.war,WEB-INF/web.xml default_host] [MicroDG1App microdg1app.war,WEB-INF/web.xml
default_host] [MicroDG2App microdg2app.war,WEB-INF/web.xml default_host] [MicroSip2App
microsip2app.war,WEB-INF/web.xml default_host]]]')
```

- **Pour configurer la gestion de session sur une application existante dans la console d'administration de WebSphere Application Server :**
  1. Dans la console d'administration de WebSphere Application Server, cliquez sur **Applications** → **Types d'application** → **Applications d'entreprise WebSphere** → *application_name* → **Web Module properties** → **Gestion des sessions** → **Paramètres de gestion des sessions**.
  2. Mettez à jour les zones pour activer la persistance des sessions dans une grille de données.

  Vous pouvez également mettre à jour l'application à l'aide d'un script wsadmin. Dans l'exemple suivant, le paramètre**-SessionManagement** crée une configuration identique à celle que vous pouvez créer dans la console d'administration :

  Pour la configuration de la grille de données eXtreme Scale éloignée :

  ```
  AdminApp.edit('DefaultApplication','
  [-SessionManagement[[[true XSRemoteSessionManagement cs0:!:grid0]]]')
  ```

  Pour le scénario imbriqué d'eXtreme Scale avec la configuration par défaut :

  ```
  AdminApp.edit('DefaultApplication','
  [-SessionManagement[[[true XSEmbeddedSessionManagement :!: :!:default]]]')
  ```

  Pour le scénario imbriqué d'eXtreme Scale avec une configuration personnalisée :

```
AdminApp.edit('DefaultApplication','
[-SessionManagement[[[true XSEmbeddedSessionManagement :!: :!:custom:!:
c:\XS\objectgrid.xml:!:c:\XS\objectgriddeployment.xml]]]')
```

Lorsque vous enregistrez les modifications, l'application utilise la grille de données configurée pour la persistance des sessions sur le dispositif.

- **Pour configurer la gestion de session sur un serveur existant :**
  1. Dans la console d'administration de WebSphere Application Server, cliquez sur **Serveurs** → **Types de serveur** → **Serveurs d'application WebSphere** → *nom_serveur* → **Paramètres du conteneur** → **Paramètres de gestion des sessions eXtreme Scale**.
  2. Mettez à jour les zones pour activer la persistance des sessions.

  Vous pouvez configurer la gestion des sessions sur un serveur existant à l'aide des commandes suivantes de l'outil wsadmin :

  Pour la configuration de la grille de données eXtreme Scale éloignée :

```
AdminTask.configureServerSessionManagement('[-nodeName
IBM-C77EE220EB6Node01
-serverName server1
-enableSessionManagement true
-sessionManagementType XSRemoteSessionManagement -XSRemoteSessionManagement
[-catalogService cs0 -csGridName grid0]]')
```

Pour la configuration imbriquée d'eXtreme Scale :

– Configuration par défaut, si vous utilisez les fichiers XML par défaut :

```
AdminTask.configureServerSessionManagement('[-nodeName IBM-C77EE220EB6Node01
-serverName server1
-enableSessionManagement true
-sessionManagementType XSEmbeddedSessionManagement

-XSEmbeddedSessionManagement [-embeddedGridType default -objectGridXML -objectGridDeploymentXML ]]')
```

– Configuration personnalisée, si vous utilisez les fichiers XML personnalisés :

```
AdminTask.configureServerSessionManagement('[-nodeName IBM-C77EE220EB6Node01
-serverName server1 -enableSessionManagement true
-sessionManagementType XSEmbeddedSessionManagement

-XSEmbeddedSessionManagement [-embeddedGridType custom -objectGridXML
c:\XS\objectgrid.xml -objectGridDeploymentXML c:\XS\objectgriddeployment.xml]]')
```

Lorsque vous enregistrez les modifications, le serveur utilise la grille de données configurée pour la persistance de sessions avec toutes les applications qu'il exécute.

Résultats

Vous avez configuré le gestionnaire de session HTTP pour qu'il stocke les sessions dans une grille de données.

ATTENTION :
Lorsque vous configurez ce scénario, les droits d'accès de sécurité d'IBM WebSphere DataPower XC10 Appliance sont automatiquement stockés dans la configuration de WebSphere Application Server. Si vous modifiez les données d'identification de la grille de données après la configuration initiale, WebSphere Application Server ne possède plus les données d'identification appropriées. Vous pouvez réinitialiser les données d'identification en réappliquant les paramètres de gestion de session eXtreme Scale.

Que faire ensuite

Fichier splicer.properties :

Le fichier splicer.properties contient toutes les options de configuration d'un gestionnaire de sessions ObjectGrid basé sur un filtre de servlet.

Exemple de fichier splicer.properties

```
# Fichier de propriétés qui contient toutes les options de configuration
# que le gestionnaire de sessions ObjectGrid basé sur un filtre de servlet
# peut être configuré pour utiliser.
#
# Ce fichier de propriétés peut être créé de sorte à conserver toutes
# les valeurs par défaut à affecter à ces paramètres de configuration
# et les paramètres individuels peuvent être remplacés à l'aide des
# propriétés de la tâche ANT, si ce fichier de propriétés est utilisé
# conjointement avec la tâche ANT filtersplicer.

# Valeur de chaîne "REMOTE" ou "EMBEDDED". La valeur par défaut est REMOTE.

# Indique si nous devons démarrer le conteneur
# WebSphere eXtreme Scale imbriqué dans tout processus de serveur d'applications
# y compris WebSphere, WebLogic, JBoss, ToimCat.

objectGridType= REMOTE

# Valeur de chaîne qui définit le nom de l'instance ObjectGrid
# utilisée pour une application Web particulière. Le nom par défaut
# est sessionmanager. La définition de ce paramètre remplace cette valeur.
# objectGridName =

# Le serveur de catalogues peut être contacté pour obtenir une instance
# ObjectGrid côté client. La valeur doit être de la forme
# "host:port<,host:port>".

# Cette liste peut être arbitrairement longue et n'est utilisée que pour l'amorçage.

# La première adresse valide est utilisée. Elle est facultative dans WebSphere
# si catalog.services.cluster est configuré

catalogHostPort = host:port<,host:port>

# Entier qui définit la durée en secondes entre
# les opérations d'écriture des sessions mises à jour dans la grille
# d'objets. La valeur par défaut est de 2 secondes.

replicationInterval = 1

# Entier qui définit le nombre de références de session conservées
# en mémoire. Elle est par défaut de 2000.

sessionTableSize =

# Valeur de chaîne "true" ou "false". La valeur par défaut est false.
# Permet de contrôler si nous stockons les données de session comme entrée
# intégrale ou si nous stockons chaque attribut séparément

fragmentedSession = false

# Indique au gestionnaire de sessions si l'utilisation du codage des URL doit
# être envisagée (à la place des cookies). La valeur par défaut
# est false

useURLEncoding = false
```

```
# Chaîne d'emplacement de fichier du fichier xml objectgrid.
# Nous chargeons automatiquement notre fichier xml pré-intégré s'il n'est
# pas spécifié et que objectGridType=EMBEDDED

objectGridXML =

# Chaîne d'emplacement de fichier du fichier xml des règles de déploiement
# d'objectGrid.
# Nous chargeons automatiquement notre fichier xml pré-intégré s'il n'est
# pas spécifié et que objectGridType=EMBEDDED

objectGridDeploymentXML =

# Chaîne de spécification de trace IBM WebShere,
# Utile pour tous les autres serveurs d'applications, tels que WebLogic.

traceSpec=

# Chaîne d'emplacement de fichier de trace.
# Utile pour tous les autres serveurs d'applications, tels que WebLogic.

traceFile=
```

Utilisation de WebSphere eXtreme Scale pour la gestion des sessions SIP :

Vous pouvez utiliser WebSphere eXtreme Scale comme mécanisme de réplication SIP (Session Initiation Protocol). Ce système se substitue en toute fiabilité au service DRS (Data Replication Service) pour la réplication des sessions SIP.

Configuration de la gestion des sessions SIP

Pour utiliser WebSphere eXtreme Scale comme mécanisme de réplication SIP, définissez la propriété personnalisée com.ibm.sip.ha.replicator.type. Dans la console d'administration, sélectionnez **Application servers** → *mon_serveur_applications* → **SIP container** → **Custom properties** pour chaque serveur auquel la propriété personnalisée doit être ajoutée. Entrez `com.ibm.sip.ha.replicator.type` pour le nom et `OBJECTGRID` pour la valeur.

Utilisez les propriétés ci-dessous pour personnaliser le comportement de la valeur ObjectGrid utilisée pour stocker les sessions SIP. Dans la console d'administration, cliquez sur **Application servers** → *mon_serveur_applications* → **SIP container** → **Custom properties** pour chaque serveur auquel la propriété personnalisée doit être ajoutée. Renseignez les champs **Nom** et **Valeur**. Les mêmes propriétés doivent être définies pour chaque serveur pour qu'il fonctionne correctement.

Tableau 10. Propriétés personnalisées pour la gestion des sessions SIP avec ObjectGrid

Propriété	Valeur	Par défaut
com.ibm.sip.ha.replicator.type	`OBJECTGRID` : utilisez ObjectGrid pour stocker les sessions SIP	
min.synchronous.replicas	Nombre minimal de fragments répliques synchrones	0
max.synchronous.replicas	Nombre maximal de fragments répliques synchrones	0
max.asynchronous.replicas	Nombre maximal de fragments répliques asynchrones	1
auto.replace.lost.shards	Pour plus d'informations, voir «Configuration de déploiements répartis», à la page 165.	true
development.mode	• `true` : active les fragments répliques sur le même noeud que les primaires • `false` : les fragments répliques doivent être sur un noeud différent que les primaires	false

Configuration du gestionnaire de sessions HTTP pour divers serveurs d'applications

WebSphere eXtreme Scale est livré avec une implémentation de la gestion des sessions qui se substitue au gestionnaire par défaut des sessions de conteneurs Web et qui fournit la réplication de sessions, la haute disponibilité, une meilleure évolutivité et de robustes options de configuration. Vous pouvez activer le gestionnaire eXtreme Scale de réplication de sessions et son démarrage générique de conteneurs ObjectGrid imbriqués.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser le gestionnaire de sessions HTTP avec d'autres serveurs d'applications qui n'exécutent pas WebSphere Application Server, WebSphere Application Server Community Edition, par exemple. Pour configurer d'autres serveurs d'applications afin qu'ils utilisent la grille de données, vous devez raccorder votre application et lui incorporer les fichiers JAR de WebSphere eXtreme Scale.

Procédure

1. Raccordez votre application de sorte qu'elle puisse utiliser le gestionnaire de sessions. Pour utiliser le gestionnaire de sessions, vous devez ajouter les déclarations de filtre appropriées aux descripteurs de déploiement Web de l'application. En outre, les paramètres de configuration du gestionnaire de sessions sont transmis au gestionnaire de sessions sous la forme de paramètres d'initialisation du contexte de servlet dans les descripteurs de déploiement. Vous disposez de trois manières de présenter ces informations dans votre application :

   - **Script addObjectGridFilter**

     Utilisez un script de ligne de commande fourni avec eXtreme Scale pour raccorder une application avec des déclarations de filtre et une configuration sous forme de paramètres d'initialisation du contexte de servlet. Ce script, `objectgridRoot/session/bin/addObjectGridFilter.sh` ou `objectgridRoot/session/bin/addObjectGridFilter.bat`, accepte deux paramètres : le chemin d'accès absolu au fichier d'archive d'entreprise de l'application que vous souhaitez raccorder, et le chemin d'accès absolu au fichier de propriétés splicer qui contient les diverses propriétés de configuration. La syntaxe de ce script est la suivante :

     Windows

     ```
     addObjectGridFilter.bat [location of ear file] [location of properties file]
     ```

     UNIX

     ```
     addObjectGridFilter.sh [location of ear file] [location of properties file]
     ```

     UNIX **Exemple d'utilisation d'eXtreme Scale installé dans un répertoire autonome sous UNIX :**

     a. `cd objectgridRoot/session/bin`

     b. `addObjectGridFilter.sh /tmp/mySessionTest.ear objectgridRoot/session/samples/splicer.properties`

     Le filtre de servlet qui est joint conserve les valeurs de configuration par défaut. Vous pouvez remplacer ces valeurs par défaut par des options de configuration que vous spécifiez dans le fichier de propriétés, dans le second argument. Pour une liste des paramètres que vous pouvez utiliser, voir «Paramètres d'initialisation du contexte de servlet», à la page 271.

Vous pouvez modifier et utiliser l'exemple de fichier `splicer.properties` fourni avec l'installation d'eXtreme Scale. Vous pouvez également utiliser le script addObjectGridServlets, qui insère le gestionnaire de sessions en étendant chaque servlet. Toutefois, le script recommandé est le script addObjectGridFilter.

- **Script de génération Ant**

  WebSphere eXtreme Scale est fourni avec un fichier `build.xml` qui peut être utilisé par Apache Ant, qui est inclus dans le dossier `wasRoot/bin` d'une installation WebSphere Application Server. Le fichier `build.xml` peut être modifié pour changer les propriétés de configuration du gestionnaire de sessions. Les propriétés de configuration sont identiques aux noms de propriété dans le fichier `splicer.properties`. Une fois que le fichier `build.xml` a été modifié, appelez le processus Ant en exécutant `ant.sh`, `ws_ant.sh` (UNIX) ou `ant.bat`, `ws_ant.bat` (Windows).

- **Modification manuelle du descripteur Web**

  Editez le fichier `web.xml` qui est packagé avec l'application Web pour incorporer la déclaration de filtre, son mappage de servlets et les paramètres d'initialisation du contexte de servlet. N'utilisez pas cette méthode car elle est source d'erreurs possibles.

Pour une liste des paramètres que vous pouvez utiliser, voir «Paramètres d'initialisation du contexte de servlet», à la page 271.

2. Incorporez dans votre application les fichiers JAR du gestionnaire de réplication de sessions d'WebSphere eXtreme Scale. Vous pouvez incorporer les fichiers dans le répertoire `WEB-INF/lib` des modules d'application ou dans le chemin d'accès aux classes du serveur d'applications. Les fichiers JAR requis varient selon le type de conteneurs utilisés :
   - conteneurs eXtreme Scale distants : `ogclient.jar` et `sessionobjectgrid.jar`
   - conteneurs eXtreme Scale imbriqués : `objectgrid.jar` et `sessionobjectgrid.jar`
3. Facultatif : Si vous utilisez des conteneurs eXtreme Scale distants, démarrez les conteneurs. Pour plus de détails, reportez-vous à la rubrique «Démarrage de processus de conteneur», à la page 333.
4. Déployez l'application. Déployez l' application à l'aide de votre procédure normale pour un serveur ou un cluster. Une fois que vous avez déployé l'application, vous pouvez la démarrer.
5. Accédez à l'application. Vous pouvez maintenant accéder à l'application, qui interagit avec le gestionnaire de sessions et WebSphere eXtreme Scale.

Que faire ensuite

Vous pouvez modifier la majorité des attributs de configuration du gestionnaire de sessions lorsque vous instrumentez votre application pour utiliser le gestionnaire de sessions. Ces attributs sont des variantes du type de réplication (synchrone ou asynchrone), la taille de la table des sessions en mémoire, etc. En dehors des attributs modifiables lors de l'instrumentation de l'application, les seuls autres attributs de configuration que vous pouvez modifier après le déploiement de l'application sont ceux liés à la topologie des clusters de serveurs WebSphere eXtreme Scale et à la manière dont leurs clients (gestionnaires de sessions) s'y connectent.

Paramètres d'initialisation du contexte de servlet

La liste qui suit de paramètres d'initialisation du contexte de servlet peut être spécifiée dans le fichier splicer.properties en fonction de la méthode de raccord choisie.

Paramètres

objectGridType

Valeur de chaîne REMOTE ou EMBEDDED. La valeur par défaut est REMOTE.

Avec REMOTE, les données de session seront stockées en dehors du serveur sur lequel s'exécute l'application Web.

Avec EMBEDDED, un conteneur eXtreme Scale démarrera dans le processus du serveur d'applications sur lequel s'exécute l'application Web.

objectGridName

Valeur de chaîne qui définit le nom de l'instance ObjectGrid utilisée pour une application Web particulière. Le nom par défaut est session.

Cette propriété doit refléter l'objectGridName qui se trouve à la fois dans le fichier XML de la grille d'objets et dans le fichier XML de déploiement utilisé pour démarrer les conteneurs eXtreme Scale.

catalogHostPort

Le serveur de catalogues peut être contacté pour obtenir une instance ObjectGrid côté client. La valeur doit être au format `hôte:port<,hôte:port>`, où hôte est l'hôte d'écoute sur lequel le serveur de catalogue s'exécute et où port est le port d'écoute pour ce processus du serveur de catalogue. La longueur de cette liste peut être arbitraire et la liste n'est utilisée que pour l'amorçage. La première adresse viable qui est utilisée. Ce paramètre est facultatif dans WebSphere Application Server si la propriété catalog.services.cluster est configurée.

replicationInterval

Entier (en secondes) qui définit le temps séparant deux écritures de sessions actualisées vers la grille. La valeur par défaut est de 2. Les valeurs possibles peuvent aller de 0 à 60. 0 signifie que les sessions actualisées sont écrites dans la grille pour chaque demande dès la fin de l'appel à la méthode de service du servlet. Une valeur supérieure à 0 améliore les performances dans la mesure où elle diminue le nombre d'écritures dans la grille. Mais, en même temps, une valeur supérieure à 0 rend la configuration moins tolérante aux pannes.

Ce paramètre s'applique uniquement lorsqu'objectGridType a la valeur REMOTE.

sessionTableSize

Entier qui définit le nombre de références de session conservées en mémoire. Elle est par défaut de 2000.

Ce paramètre ne concerne qu'une topologie REMOTE car la topologie EMBEDDED dispose déjà des données de session dans le même groupe que le conteneur Web.

Les sessions sont expulsées de la table en mémoire sur une base d'utilisations les moins récentes. Lorsqu'une session est expulsée de cette table, elle est invalidée du conteneur Web mais les données ne sont pas pour autant supprimées de la grille, ce qui permet aux demandes ultérieures de cette session de continuer à extraire les données.

fragmentedSession

Valeur de chaîne true ou false. La valeur par défaut est true. Ce paramètre permet de contrôler si le produit stocke les données de session en tant qu'entrée entière ou s'il stocke chaque attribut séparément.

Donnez à fragmentedSession la valeur true si la session de l'application Web comporte un grand nombre d'attributs ou des attributs de taille importante. Ne lui donnez la valeur false que si la session comporte peu d'attributs, car tous les attributs sont stockés dans la même clé dans la grille.

Dans la précédente implémentation à base de filtres, il était fait référence à cette propriété en tant que mécanisme de persistance avec, comme valeurs possibles, ObjectGridStore (fragmentation) et ObjectGridAtomicSessionStore (non-fragmentation).

securityEnabled

Valeur de chaîne true ou false. La valeur par défaut est false. Ce paramètre active la sécurité du client eXtreme Scale. Il doit également correspondre au paramètre securityEnabled du fichier des propriétés des serveurs eXtreme Scale. Si les paramètres ne correspondent pas, une exception est générée.

credentialGeneratorClass

Le nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Cette classe sert à obtenir les données d'identification des clients.

credentialGeneratorProps

Les propriétés de la classe d'implémentation CredentialGenerator. Les propriétés correspondent à l'objet avec la méthode **setProperties(String)**. La valeur credentialGeneratorProps n'est utilisée que si la valeur de la propriété **credentialGeneratorClass** n'est pas null.

objectGridXML

L'emplacement du fichier `objectgrid.xml`. Le fichier XML pré-intégré qui est packagé dans la bibliothèque eXtreme Scale sera chargé automatiquement si objectGridType=EMBEDDED et si la propriété **objectGridXML** n'est pas spécifiée.

objectGridDeploymentXML

L'emplacement du fichier XML de la règle de déploiement de la grille d'objets. Le fichier XML pré-intégré qui est packagé dans la bibliothèque eXtreme Scale sera chargé automatiquement si objectGridType=EMBEDDED et si la propriété **objectGridDeploymentXML** n'est pas spécifiée.

traceSpec

La spécification de trace d'IBM WebSphere en tant que valeur de chaîne. Utilisez ce paramètre pour des serveurs d'applications autres que WebSphere Application Server.

traceFile

L'emplacement du fichier de trace, en tant que valeur de chaîne. Utilisez ce paramètre pour des serveurs d'applications autres que WebSphere Application Server.

Configuration du fournisseur de cache dynamique pour WebSphere eXtreme Scale

L'installation et la configuration du fournisseur de cache dynamique d'eXtreme Scale varient en fonction de vos exigences et de l'environnement que vous avez configuré.

Avant de commencer

Installez le fournisseur de cache dynamique.

Pour pouvoir utiliser le fournisseur de cache dynamique, WebSphere eXtreme Scale doit être installé par dessus les déploiements de noeud WebSphere Application Server et notamment le noeud du gestionnaire de déploiement. Le fournisseur de cache dynamique d'eXtreme Scale est pris en charge sur les versions ci-après de WebSphere Application Server.

- WebSphere Application Server 6.1.0.25 + PK85622 et versions ultérieures
- WebSphere Application Server 7.0.0.3 + PK85622 et versions ultérieures

Pour des instructions d'installation, voir la rubrique relatives à l'installation pour la version 6.1 ou à l'installation pour la version 7.0.

Pour savoir comment utiliser le fournisseur de cache dynamique d'eXtreme Scale avec IBM WebSphere Commerce, voir les rubriques suivantes dans la documentation d'IBM WebSphere Commerce :

- Activation du service de cache dynamique et de la mise en cache des servlets
- Activation du cache de données de WebSphere Commerce

Pourquoi et quand exécuter cette tâche

Pour configurer le fournisseur de cache dynamique d'eXtreme Scale, suivez ces étapes :

1. Activez le fournisseur de cache dynamique d'eXtreme Scale.

   Dans WebSphere Application Server 7.0, vous pouvez configurer le service de cache dynamique pour utiliser le fournisseur de cache dynamique eXtreme Scale avec la console d'administration ou une propriété personnalisée.

   Une fois que vous avez installé eXtreme Scale, le fournisseur de cache dynamique eXtreme Scale est immédiatement disponible comme option de fournisseur de cache dynamique dans la console d'administration. Pour plus d'informations, voir Sélection d'un fournisseur de service de cache.

   Vous pouvez également configurer le fournisseur de cache dynamique d'eXtreme Scale pour une instance de cache en définissant les paires de propriété personnalisée et de valeur ci-après sur l'instance. Ces propriétés personnalisées représentent le seul moyen d'activer le fournisseur eXtreme Scale sur WebSphere Application Server 6.1, comme suit :

   ```
   com.ibm.ws.cache.CacheConfig.cacheProviderName =
      "com.ibm.ws.objectgrid.dynacache.CacheProviderImpl"
   ```

   Si vous ne dirigez pas spécifiquement votre mise en cache vers une instance ObjectCache ou ServletCache définie, il y a de fortes chances que les appels à l'API DynamicCache soient traités par baseCache, le cache de base. baseCache est l'instance de cache par défaut créée dans le cadre du service DynamicCache. Les mêmes propriétés de configuration permettent de configurer l'instance baseCache pour utiliser eXtreme Scale comme son fournisseur de cache. Mais

ces propriétés de configuration ont besoin d'être définies comme propriétés personnalisées de machine virtuelle Java pour qu'elles puissent affecter le cache de base.

Définir des variables de configuration en tant que propriété personnalisée de machine virtuelle Java peut avoir pour effet que d'autres caches les sélectionnent également. Les instances ObjectCache et ServletCache doivent préférer les propriétés personnalisées définies dans l'instance de cache mais, si vous constatez qu'une instance de cache spécifique utilise le fournisseur eXtreme Scale alors qu'elle devrait utiliser le fournisseur par défaut, il est possible de pallier le problème à l'aide des propriétés personnalisées. Pour obliger une instance de cache à utiliser le fournisseur de cache dynamique par défaut, définissez comme suit la propriété de fournisseur de cache:

```
com.ibm.ws.cache.CacheConfig.cacheProviderName = "default"
```

Les propriétés de configuration du fournisseur de cache dynamique d'eXtreme Scale définies pour baseCache pouvant être sélectionnées par d'autres instances de cache, il ne faut s'appuyer qu'à bon escient sur les valeurs par défaut lorsqu'on définit des propriétés de configuration du cache.

2. Configurez le paramètre de réplication du cache.

   Avec le fournisseur de cache dynamique eXtreme Scale, il est possible d'avoir des instances de cache local et des des instances de cache répliqué. Dans le cas des instances de cache local, aucune autre configuration n'est requise et le reste de cette section peut être ignoré.

   Les caches répliqués peuvent être créés de deux manières différentes. La première consiste à activer la réplication du cache via la console d'administration. Cette activation peut être effectuée à tout moment dans WebSphere Application Server version 7.0, mais nécessite de créer un domaine de réplication DRS dans WebSphere Application Server version 6.1.

   La deuxième consiste à utiliser la propriété personnalisée et la paire de valeurs ci-après pour forcer le cache à signaler qu'il est un cache répliqué, bien qu'un domaine de réplication DRS ne lui ait pas été affecté.

   ```
   com.ibm.ws.cache.CacheConfig.enableCacheReplication = "true"
   ```

3. Configurez la topologie pour le service de cache dynamique.

   Le seul paramètre de configuration requis pour le fournisseur de cache dynamique d'eXtreme Scale est la topologie du cache. La variable ci-après doit être définie comme propriété personnalisée sur le service de cache dynamique.

   ```
   com.ibm.websphere.xs.dynacache.topology
   ```

   Les trois valeurs possibles pour cette propriété sont les suivantes :

   - embedded
   - embedded_partitioned
   - remote

   Vous devez utiliser l'une des valeurs autorisées.

4. Configurez le nombre de conteneurs initiaux pour le service de mise en cache dynamique.

   Vous pouvez maximiser les performances des mémoires cache qui utilisent la topologie partitionnée imbriquée en configurant le nombre initial de conteneurs. La variable ci-après doit être définie comme propriété système sur la machine virtuelle Java de WebSphere Application Server.

   ```
   com.ibm.websphere.xs.dynacache.num_initial_containers
   ```

   La valeur recommandée pour cette propriété de configuration est un entier égal ou légèrement inférieur au nombre total d'instances WebSphere Application

Server qui accèdent à cette instance de cache réparti. Par exemple, si un service de cache dynamique est partagé entre les membres d'une grille, la valeur de la variable doit correspondre au nombre de membres de la grille.

Pour des topologies imbriquées ou partitionnées imbriquées, vous devez utiliser la version 7.0 de WebSphere Application Server. Définissez la propriété suivante dans le processus JVM pour vous assurer que les conteneurs initiaux sont immédiatement disponibles.

```
com.ibm.ws.cache.CacheConfig.createCacheAtServerStartup=true
```

5. Configurez la grille du service de catalogue d'eXtreme Scale.

   Lorsque vous utilisez eXtreme Scale comme le fournisseur de mémoire cache dynamique pour une instance de cache réparti, vous devez configurer un domaine de services de catalogue eXtreme Scale.

   Un même domaine de services de catalogue peut traiter plusieurs fournisseurs de services de cache dynamique s'appuyant sur eXtreme Scale.

   7.1+ Un service de catalogue peut être exécuté à l'intérieur ou à l'extérieur des processus WebSphere Application Server. A partir de la version 7.1 d'eXtreme Scale, lorsqu'on utilise la console d'administration afin de configurer des mécanismes de domaine pour des serveurs de catalogue, le cache dynamique utilisera ces paramètres. Il n'est pas nécessaire de procéder à une configuration supplémentaire pour définir un service de catalogue. Pour plus d'informations, voir «Création de domaines de service de catalogue dans WebSphere Application Server», à la page 346. Lorsque vous exécutez un domaine de services de catalogue, vous devez définir la propriété personnalisée **catalog.services.cluster** pour les points de contact des services de catalogue.

6. Configurez les objets de clé personnalisés.

   Si vous utilisez des objets personnalisés comme des clés, ces objets doivent implémenter l'interface Serializable ou Externalizable. Lorsque vous utilisez des topologies imbriquées ou des topologies partitionnées imbriquées, vous devez placer les objets dans le chemin d'accès à la bibliothèque partagée de WebSphere, comme s'ils étaient utilisés avec le fournisseur de cache dynamique par défaut. Pour plus de détails, voir Utilisation des interfaces DistributedMap et DistributedObjectCache pour le cache dynamique dans le Centre de documentation de WebSphere Application Server Network Deployment.

   Si vous utilisez la topologie éloignée, vous devez placer les objets de clé personnalisés dans le chemin d'accès aux classes (CLASSPATH) des conteneurs eXtreme Scale autonomes.

7. Configurez les serveurs conteneurs d'eXtreme Scale.

   Les données en cache sont stockées dans des conteneurs WebSphere eXtreme Scale. Les conteneurs peuvent être exécutés à l'intérieur ou à l'extérieur des processus WebSphere Application Server. Le fournisseur eXtreme Scale crée automatiquement des conteneurs dans le processus WebSphere si vous utilisez des topologies imbriquées ou partitionnées imbriquées pour une instance de cache. Aucune configuration supplémentaire n'est requise pour ces topologies.

   Si vous utilisez la topologie éloignée, vous devez démarrer les conteneurs eXtreme Scale autonomes avant que les instances WebSphere Application Server qui accèdent à l'instance de cache ne démarrent. Vérifiez que tous les conteneurs d'un service de cache dynamique spécifique pointent vers les mêmes points de contact de service de catalogue.

   Les fichiers de configuration XML des conteneurs autonomes du fournisseur de cache dynamique d'eXtreme Scale se trouvent dans le répertoire `<racine_install>/customLibraries/ObjectGrid/dynacache/etc` pour les installations effectuées par dessus WebSphere Application Server ou dans le

répertoire <racine_install>/ObjectGrid/dynacache/etc pour les installations autonomes. Les fichiers s'intitulent dynacache-remote-objectgrid.xml et dynacache-remote-definition.xml. Effectuez des copies de ces fichiers pour les éditer et les utiliser lors du lancement des conteneurs autonomes du fournisseur de cache dynamique d'eXtreme Scale. Le paramètre **numIntitialContainers** du fichier **dynacache-remote-deployment.xml** doit être défini en conséquence avec le nombre de processus de conteneur exécuté. Notez que l'attribut **numberOfPartitions** dans le fichier dynacache-remote-objectgrid.xml a par défaut une valeur de 47.

Remarque : L'ensemble des processus de conteneur doit disposer d'une quantité de mémoire disponible suffisante pour traiter toutes les instances de cache dynamique configurées pour utiliser la topologie éloignée. Tout processus WebSphere Application Server qui partage les mêmes valeurs ou des valeurs équivalentes de **catalog.services.cluster** doit utiliser le même ensemble de conteneurs autonomes. Le nombre de conteneurs et le nombre de machines sur lesquelles ils se trouvent doivent être définis de manière appropriée. Pour plus de détails, voir «Planification de la capacité et haute disponibilité (mise en cache dynamique)», à la page 13.

Un exemple d'entrée de ligne de commande UNIX qui lance un conteneur autonome pour le fournisseur de cache dynamique d'eXtreme Scale est présenté dans le code suivant :

```
startOgServer.sh container1 -objectGridFile ../dynacache/etc/dynacache-remote-objectgrid.xml -deploymentPolicyFile ../dynacache/etc/dynacache-remote-deployment.xml -catalogServiceEndpoints MyServer1.company.com:2809
```

8. Dans le cas de topologies réparties ou imbriquées, activez l'agent de dimensionnement pour améliorer les estimations d'utilisation de la mémoire.

   L'agent de dimensionnement estime l'utilisation qui sera faite de la mémoire (statistiques usedBytes). L'agent requiert Java 5 ou supérieur.

   Chargez l'agent en ajoutant l'argument suivant à la ligne de commande de la machine virtuelle Java :

   ```
   -javaagent:rép_biblio_WXS/wxssizeagent.jar
   ```

   Dans le cas d'une topologie imbriquée, ajoutez l'argument à la ligne de commande du processus WebSphere Application Server.

   Dans le cas d'une topologie répartie, ajoutez l'argument à la ligne de commande des processus eXtreme Scale (conteneurs) et du processus WebSphere Application Server.

Configuration de l'intégration à Spring

Fichier XML du descripteur de Spring

Utilisez un fichier XML de descripteur de Spring pour configurer et intégrer eXtreme Scale à Spring.

Dans les sections ci-après, chacun des éléments et attributs du fichier objectgrid.xsd de Spring est défini. Le fichier objectgrid.xsd de Spring se trouve dans le fichier ogspring.jar et l'espace de noms com/ibm/ws/objectgrid/spring/namespace d'objectgrid. Pour un exemple de schéma XML de descripteur, voir la rubrique «Fichier objectgrid.xsd de Spring», à la page 283.

Elément register

Utilisez l'élément register pour enregistrer la fabrique de beans par défaut de l'ObjectGrid.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

id Spécifie le nom du répertoire de beans par défaut d'un ObjectGrid particulier.

gridname
Spécifie le nom de l'instance ObjectGrid. La valeur affectée à cet attribut doit correspondre à un ObjectGrid valide configuré dans le fichier du descripteur d'ObjectGrid.

```
<register
(1) id="register id"
(2) gridname="ObjectGrid name"
/>
```

Elément server

L'élément server permet de définir un serveur eXtreme Scale, qui peut héberger un conteneur et/ou un service de catalogue.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

id Spécifie le nom du serveur eXtreme Scale.

tracespec
Indique le type de trace et active la trace et la spécification de trace pour le serveur.

tracefile
Spécifie le chemin d'accès et le nom du fichier de trace à créer et utiliser.

statspec
Indique la spécification des statistiques du serveur.

jmxport
Désigne le numéro de port inutilisé par l'intermédiaire duquel vous souhaitez activer les connexions JMX/RMI. JMX permet la surveillance et la gestion à partir de systèmes éloignés.

isCatalog
Spécifie si le serveur particulier héberge un service de catalogue. La valeur par défaut est `false`.

name
Spécifie le nom du serveur.

haManagerPort
Définit le numéro du port pour le gestionnaire de haute disponibilité (HA Manager).

listenerHost
Définit le nom de l'hôte auquel l'ORB doit se lier.

listenerPort
Définit le port auquel l'ORB doit se lier.

maximumThreadPoolSize
Définit le nombre maximum d'unités d'exécution dans le pool.

memoryThresholdPercentage
Définit le seuil (pourcentage du segment maximum) pour l'expulsion en fonction de la mémoire.

minimumThreadPoolSize
Définit le nombre minimum d'unités d'exécution dans le pool.

workingDirectory
La propriété qui définit le répertoire que doit utiliser le serveur ObjectGrid pour tous ses paramètres par défaut.

zoneName
Définit la zone à laquelle appartient ce serveur.

enableChannelFramework
Définit ChannelFramework comme le mode de transport dans les propriétés de l'IBM ORB.

enableSystemStreamToFile
Définit si SystemOut et SystemErr doivent être ou non envoyés à un fichier.

enableMBeans
Détermine si la grille d'objets enregistrera ou non des beans gérés dans ce processus.

serverPropertyFile
Charge les propriétés de serveur à partir d'un fichier.

catalogServerProperties
Spécifie le serveur de catalogue qui héberge le serveur.

```
<server
(1) id="server id"
(2) tracespec="la spécification de trace du serveur"
(3) tracefile="le fichier de trace du serveur"
(4) statspec="la spécification des statistiques du serveur"
(5) jmxport="le numéro de port JMX"
(6) isCatalog="true"|"false"
(7) name="le nom du serveur"
(8) haManagerPort="le port du gestionnaire de haute disponibilité"
(9) listenerHost="le nom d'hôte de la liaison ORB"
(10) listenerPort="le port d'écoute de la liaison ORB"
(11) maximumThreadPoolSize="le nombre maximum d'unités d'exécution"
(12) memoryThresholdPercentage="le seuil mémoire (pourcentage du segment maximum)"
(13) minimumThreadPoolSize="le nombre minimum d'unités d'exécution"
(14) workingDirectory="l'emplacement du répertoire de travail"
(15) zoneName="le nom de la zone"
(16) enableChannelFramework="true"|"false"
(17) enableSystemStreamToFile="true"|"false"
(18) enableMBeans="true"|"false"
(19 serverPropertyFile="l'emplacement du fichier de propriétés du serveur"
(20) catalogServerProperties="la référence aux propriétés du serveur de catalogue"
/>
```

Elément catalog

L'élément catalog permet d'acheminer vers des serveurs conteneurs dans la grille de données.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

host
Spécifie le nom d'hôte du poste de travail sur lequel le service de catalogue est exécuté.

port
Spécifie le numéro de port associé au nom d'hôte pour déterminer le port du service de catalogue auquel le client peut se connecter.

```
<catalog
(1) host="catalog service host name"
(2) port="catalog service port number"
/>
```

Elément catalog server

L'élément de propriétés catalog server permet de définir un service de serveur de catalogue.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

catalogServerEndPoint
Spécifie les propriétés de connexion du serveur de catalogue.

enableQuorum
Détermine si le quorum doit être activé.

heartBeatFrequencyLevel
Définit le niveau de fréquence des signaux de présence.

domainName
Définit le nom de domaine utilisé afin d'identifier de manière exclusive cette grille de services de catalogue pour les clients lors d'un routage vers plusieurs domaines.

foreignDomains
Une connexion bidirectionnelle sera établie vers chacun des domaines externes dans le but d'échanger des données dans un scénario à multiples serveurs primaires.

clusterSecurityURL
Définit l'emplacement du fichier de sécurité spécifique au service de catalogue.

```
<catalog server
(1) catalogServerEndPoint="référence à des points de contact de serveur de catalogue"
(2) enableQuorum="true"|"false"
(3) heartBeatFrequencyLevel="
  HEARTBEAT_FREQUENCY_LEVEL_TYPICAL|
  HEARTBEAT_FREQUENCY_LEVEL_RELAXED|
  HEARTBEAT_FREQUENCY_LEVEL_AGGRESSIVE"
(4) domainName="le nom du domaine utilisé pour identifier de manière exclusive cette grille de
    service de catalogue"
(5) domainEndpoints="référence à des points de contacts du nom de domaine"
(6) foreignDomains="le nom du domaine externe"
(7) clusterSecurityURL="l'emplacement du fichier de sécurité du cluster"/>
```

Elément catalog server endpoint

L'élément catalog server endpoint permet de créer un point de contact de serveur de catalogue utilisable par un élément catalog server.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

serverName
Spécifie le nom qui permettra d'identifier le processus que vous lancez.

hostName
Spécifie le nom d'hôte de l'ordinateur sur lequel le serveur est lancé.

clientPort
Spécifie le port utilisé pour les communications homologues de cluster de catalogues.

peerPort
Spécifie le port utilisé pour les communications homologues de cluster de catalogues.

```
<catalogServerEndPoint
(1) name="le nom du point de contact de serveur de catalogue"
(2) host=""
(3) clientPort=""
(4) peerPort""
/>
```

Elément container

L'élément container permet de stocker les données.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

objectgridxml
Spécifie le chemin d'accès et le nom du fichier XML du descripteur à utiliser qui spécifie les caractéristiques de l'ObjectGrid, avec les mappes, la stratégie de verrouillage et les plug-in.

deploymentxml
Spécifie le chemin d'accès et le nom du fichier XML utilisé avec le XML du descripteur pour déterminer le partitionnement, la réplication, le nombre de conteneurs initial et d'autres paramètres.

server
Spécifie le serveur sur lequel le conteneur est hébergé.

```
<server
(1) objectgridxml="fichier XML du descripteur d'objectgrid"
(2) deploymentxml ="fichier XML du descripteur de déploiement d'objectgrid"
(3) server="référence du serveur "
/>
```

Elément JPALoader

Utilisez l'élément JPALoader pour synchroniser le cache d'ObjectGrid avec un fichier de données dorsal existant si vous utilisez l'API ObjectMap.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

entityClassName
Permet l'utilisation d'API de persistance Java, telles qu'EntityManager.persist et EntityManager.find. L'attribut **entityClassName** est requis pour le chargeur JPA.

preloadPartition
Spécifie le numéro de partition à partir duquel le préchargement de la mappe démarre. Si la valeur est inférieure à 0 ou supérieure à (totalNumberOfPartition – 1), le préchargement de la mappe n'est pas démarré.

```
<JPALoader
(1) entityClassName="nom de la classe d'entités"
(2) preloadPartition ="int"
/>
```

Elément JPATxCallback

Utilisez l'élément JPATxCallback pour coordonner les transactions JPA et ObjectGrid.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

persistenceUnitName
Crée une fabrique de gestionnaire d'entités JPA et recherche les métadonnées des entités JPA dans le fichier persistence.xml. L'attribut **persistenceUnitName** est requis.

jpaPropertyFactory
Spécifie la fabrique permettant de créer une mappe de propriétés de persistance pour remplacer les propriétés de persistance par défaut. Cet attribut doit faire référence à un bean.

exceptionMapper
Spécifie le plug-in ExceptionMapper qui peut être utilisé pour les fonctions de mappage des exceptions spécifiques à JPA ou à la base de données. Cet attribut doit faire référence à un bean.

```
<JPATxCallback
(1) persistenceUnitName="nom de l'unité de persistance JPA"
(2) jpaPropertyFactory ="référence du bean JPAPropertyFactory"
(3) exceptionMapper="référence du bean ExceptionMapper"
/>
```

Elément JPAEntityLoader

Utilisez l'élément JPAEntityLoader pour synchroniser le cache d'ObjectGrid avec un fichier de données dorsal existant si vous utilisez l'API EntityManager.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

entityClassName
Permet l'utilisation d'API de persistance Java, telles qu'EntityManager.persist et EntityManager.find. L'attribut **entityClassName** est facultatif pour l'élément JPAEntityLoader. Si l'élément n'est pas configuré, la classe d'entités configurée dans la mappe d'entités d'ObjectGrid est utilisée. La même classe doit être utilisée pour le gestionnaire d'entités d'ObjectGrid et le fournisseur JPA.

preloadPartition
Spécifie le numéro de partition à partir duquel le préchargement de la mappe démarre. Si la valeur est inférieure à 0 ou supérieure à (totalNumberOfPartition – 1), le préchargement de la mappe n'est pas lancé.

```
<JPAEntityLoader
(1) entityClassName="nom de la classe d'entités"
(2) preloadPartition ="int"
/>
```

Elément LRUEvictor

L'élément LRUEvictor permet de déterminer les entrées à expulser lorsqu'une mappe dépasse son nombre d'entrées maximal.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

maxSize
Spécifie le nombre total d'entrées dans une file d'attente avant que l'expulseur ne doive intervenir.

sleepTime
Définit la durée en secondes entre le balayage des files d'attente de la mappe par l'expulseur pour déterminer les éventuelles actions requises sur la mappe.

numberOfLRUQueues
Spécifie le nombre de files d'attente que l'expulseur doit analyser pour éviter d'avoir une seule file d'attente dont la taille correspond à celle de la mappe toute entière.

```
<LRUEvictor
(1) maxSize="int"
(2) sleepTime ="secondes"
(3) numberOfLRUQueues ="int"
/>
```

Elément LFUEvictor

L'élément LFUEvictor permet de déterminer les entrées à expulser lorsqu'une mappe dépasse son nombre d'entrées maximal.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

maxSize
Spécifie le nombre total d'entrées autorisées dans chaque segment de mémoire avant que l'expulseur ne doive intervenir.

sleepTime
Définit la durée en secondes entre le balayage des segments de mémoire de la mappe par l'expulseur pour déterminer les éventuelles actions requises sur la mappe.

numberOfHeaps
Spécifie le nombre de segments de mémoire que l'expulseur doit analyser pour éviter d'avoir un seul segment de mémoire dont la taille correspond à celle de la mappe toute entière.

```
<LFUEvictor
(1) maxSize="int"
(2) sleepTime ="secondes"
(3) numberOfHeaps ="int"
```

Elément HashIndex

Utilisez l'élément HashIndex avec la réflexion Java pour introspecter de manière dynamique les objets stockés dans une mappe lorsque les objets sont mis à jour.

- nombre d'occurrences : zéro à plusieurs
- élément enfant : aucun

Attributs

name
Spécifie le nom de l'index, qui doit être unique pour chaque mappe.

attributeName
Spécifie le nom de l'attribut à indexer. Pour les index d'accès par champ, le nom d'attribut est équivalent au nom de champ. Pour les index d'accès par propriété, le nom d'attribut correspond au nom de la propriété compatible JavaBean.

rangeIndex
Indique si l'indexation des plages est activée. La valeur par défaut est `false`.

fieldAccessAttribute
Utilisée pour les mappes qui ne correspondent pas à des entités. La méthode d'accès get permet d'accéder aux données. La valeur par défaut est `false`. Si vous spécifiez la valeur `true`, l'objet est accessible directement par les champs.

POJOKeyIndex
Utilisée pour les mappes qui ne correspondent pas à des entités. La valeur par défaut est `false`. Si vous spécifiez la valeur `true`, l'index introspecte l'objet dans la partie clé de la mappe, ce qui est utile si la clé est une clé composite et que la valeur ne contient pas la clé. Si vous ne spécifiez pas de valeur ou que vous spécifiez la valeur `false`, l'index introspecte l'objet dans la partie valeur de la mappe.

```
<HashIndex
(1) name="nom de l'index"
(2) attributeName="nom de l'attribut"
(3) rangeIndex ="true"|"false"
(4) fieldAccessAttribute ="true"|"false"

(5) POJOKeyIndex ="true"|"false"
/>
```

Fichier `objectgrid.xsd` de Spring

Utilisez le fichier `objectgrid.xsd` de Spring pour intégrer eXtreme Scale à Spring afin de gérer les transactions eXtreme Scale et configurer les clients et serveurs.

Pour les descriptions des éléments et des attributs définis dans le fichier `objectgrid.xsd` de Spring, reportez-vous à la rubrique «Fichier XML du descripteur de Spring», à la page 276.

Fichier `objectgrid.xsd` de Spring

```
<xsd:schema xmlns="http://www.ibm.com/schema/objectgrid"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:beans="http://www.springframework.org/schema/beans"
    targetNamespace="http://www.ibm.com/schema/objectgrid"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">

   <xsd:import namespace="http://www.springframework.org/schema/beans" />

 <xsd:element name="transactionManager">
      <xsd:complexType>
         <xsd:attribute name="id" type="xsd:ID" />
      </xsd:complexType>
   </xsd:element>

 <xsd:element name="register">
      <xsd:complexType>
         <xsd:attribute name="id" type="xsd:ID" />
         <xsd:attribute name="gridname" type="xsd:string" />
      </xsd:complexType>
   </xsd:element>

 <xsd:element name="server">
      <xsd:complexType>
```

```
 <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element ref="catalog" />
       </xsd:choice>
       <xsd:attribute name="id" type="xsd:ID" />
       <xsd:attribute name="tracespec" type="xsd:string" />
       <xsd:attribute name="tracefile" type="xsd:string" />
       <xsd:attribute name="statspec" type="xsd:string" />
       <xsd:attribute name="jmxport" type="xsd:integer" />
       <xsd:attribute name="isCatalog" type="xsd:boolean" />
       <xsd:attribute name="name" type="xsd:string" />
       <xsd:attribute name="haManagerPort" type="xsd:integer"/>
       <xsd:attribute name="listenerHost" type="xsd:string"/>
       <xsd:attribute name="listenerPort" type="xsd:integer"/>
       <xsd:attribute name="maximumThreadPoolSize" type="xsd:integer"/>
       <xsd:attribute name="memoryThresholdPercentage" type="xsd:integer"/>
       <xsd:attribute name="minimumThreadPoolSize" type="xsd:integer"/>
       <xsd:attribute name="workingDirectory" type="xsd:string"/>
       <xsd:attribute name="zoneName" type="xsd:string"/>
       <xsd:attribute name="enableChannelFramework" type="xsd:boolean"/>
       <xsd:attribute name="enableSystemStreamToFile" type="xsd:boolean"/>
       <xsd:attribute name="enableMBeans" type="xsd:boolean"/>
       <xsd:attribute name="serverPropertyFile" type="xsd:string"/>
       <xsd:attribute name="catalogServerProperties" type="xsd:string"/>
    </xsd:complexType>
</xsd:element>

<xsd:element name="catalog">
  <xsd:complexType>
    <xsd:attribute name="host" type="xsd:string" />
    <xsd:attribute name="port" type="xsd:integer" />
  </xsd:complexType>
</xsd:element>

<xsd:element name="catalogServerProperties">
  <xsd:complexType>
 <xsd:choice minOccurs="0" maxOccurs="unbounded">
          <xsd:element ref="catalogServerEndPoint"/>
       </xsd:choice>
      <xsd:attribute name="id" type="xsd:ID"/>
    <xsd:attribute name="enableQuorum" type="xsd:boolean"/>
    <xsd:attribute name="heartBeatFrequencyLevel" type="xsd:integer"/>
    <xsd:attribute name="domainName" type="xsd:string"/>
    <xsd:attribute name="domainEndpoints" type="xsd:string"/>
    <xsd:attribute name="foreignDomains" type="xsd:string"/>
    <xsd:attribute name="clusterSecurityURL" type="xsd:anyURI"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="catalogServerEndPoint">
  <xsd:complexType>
     <xsd:attribute name="name" type="xsd:string" />
    <xsd:attribute name="host" type="xsd:string" />
    <xsd:attribute name="clientPort" type="xsd:integer"/>
    <xsd:attribute name="peerPort" type="xsd:integer"/>
  </xsd:complexType>
</xsd:element>

<xsd:element name="container">
     <xsd:complexType>
        <xsd:attribute name="id" type="xsd:ID" />
        <xsd:attribute name="objectgridxml" type="xsd:string" />
        <xsd:attribute name="deploymentxml" type="xsd:string" />
        <xsd:attribute name="server" type="xsd:string" />
     </xsd:complexType>
  </xsd:element>

   <xsd:element name="JPALoader">
     <xsd:complexType>
        <xsd:attribute name="id" type="xsd:ID" />
        <xsd:attribute name="entityClassName" type="xsd:string" />
        <xsd:attribute name="preloadPartition" type="xsd:integer" />
     </xsd:complexType>
  </xsd:element>

  <xsd:element name="JPATxCallback">
     <xsd:complexType>
        <xsd:attribute name="id" type="xsd:ID" />
        <xsd:attribute name="persistenceUnitName" type="xsd:string" />
  <xsd:attribute name="jpaPropertyFactory" type="xsd:string" />
  <xsd:attribute name="exceptionMapper" type="xsd:string" />
  </xsd:complexType>
  </xsd:element>

  <xsd:element name="JPAEntityLoader">
     <xsd:complexType>
        <xsd:attribute name="id" type="xsd:ID" />
        <xsd:attribute name="entityClassName" type="xsd:string" />
        <xsd:attribute name="preloadPartition" type="xsd:integer" />
     </xsd:complexType>
  </xsd:element>
```

```
   <xsd:element name="LRUEvictor">
      <xsd:complexType>
         <xsd:attribute name="id" type="xsd:ID" />
         <xsd:attribute name="maxSize" type="xsd:integer" />
         <xsd:attribute name="sleepTime" type="xsd:integer" />
         <xsd:attribute name="numberOfLRUQueues" type="xsd:integer" />
         <xsd:attribute name="useMemoryUsageThresholdEviction" type="xsd:boolean" />
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="LFUEvictor">
      <xsd:complexType>
         <xsd:attribute name="id" type="xsd:ID" />
         <xsd:attribute name="maxSize" type="xsd:integer" />
         <xsd:attribute name="sleepTime" type="xsd:integer" />
         <xsd:attribute name="numberOfHeaps" type="xsd:integer" />
         <xsd:attribute name="useMemoryUsageThresholdEviction" type="xsd:boolean" />
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="HashIndex">
      <xsd:complexType>
         <xsd:attribute name="id" type="xsd:ID" />
         <xsd:attribute name="name" type="xsd:string" />
         <xsd:attribute name="attributeName" type="xsd:string" />
         <xsd:attribute name="rangeIndex" type="xsd:boolean" />
         <xsd:attribute name="fieldAccessAttribute" type="xsd:boolean" />
         <xsd:attribute name="POJOKeyIndex" type="xsd:boolean" />
      </xsd:complexType>
   </xsd:element>

</xsd:schema>
```

Prise en charge des beans d'extension Spring et des espaces de noms

WebSphere eXtreme Scale fournit une fonction permettant de déclarer des objets Java simples (POJO) afin de les utiliser en tant que points d'extension dans le fichier `objectgrid.xml` et le moyen de nommer les beans, puis de spécifier le nom de classe. En règle générale, les instances de la classe spécifiée sont créées et ces objets sont utilisés en tant que plug-in. Maintenant, eXtreme Scale peut déléguer à Spring l'obtention d'instances de ces objets de plug-in. Si une application utilise Spring, alors de tels objets Java simples doivent être connectés au reste de l'application.

Dans certains cas, vous devez utiliser Spring pour configurer certains objets de plug-in. Prenez l'exemple de la configuration suivante :

```
<objectGrid name="Grid">
    <bean id="TransactionCallback" className="com.ibm.websphere.objectgrid.jpa.JPATxCallback">
        <property name="persistenceUnitName" type="java.lang.String"  value="employeePU" />
    </bean>
    ...
</objectGrid>
```

L'implémentation pré-intégrée de TransactionCallback, classe com.ibm.websphere.objectgrid.jpa.JPATxCallback, est configurée comme classe TransactionCallback. Cette classe est configurée avec la propriété persistenceUnitName (voir l'exemple précédent). La classe JPATxCallback comporte également l'attribut JPAPropertyFactory attribute, qui est de type java.lang.Object. La configuration ObjectGrid XML ne prend pas ce type de configuration en charge.

L'intégration de eXtreme Scale Spring résout ce problème en déléguant à Spring la création des beans. La configuration révisée est comme suit :

```
<objectGrid name="Grid">
    <bean id="TransactionCallback" className="{spring}jpaTxCallback"/>
    ...
</objectGrid>
```

Le fichier Spring pour l'objet "Grid" comporte les informations suivantes :

```
<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>
```

Ici, TransactionCallback est spécifié comme `{spring}jpaTxCallback` et les beans jpaTxCallback et jpaPropFactory sont configurés dans le fichier Spring comme indiqué dans l'exemple précédent. La configuration Spring permet de configurer un bean JPAPropertyFactory en tant que paramètre de l'objet JPATxCallback.

Classe de bean Spring par défaut

Lorsque eXtreme Scale détecte un plug-in ou un bean d'extension (par exemple, ObjectTransformer, Loader ou TransactionCallback etc.) avec une valeur de nom de classe dotée du préfixe `{spring}`, alors eXtreme Scale utilise le reste du nom en tant que nom de bean Spring et obtient l'instance de bean à l'aide de la classe Spring Bean.

Par défaut, si aucune classe de bean n'a été enregistrée pour un objet ObjectGrid donné, il tente alors de trouver un fichier `ObjectGridName_spring.xml`. Par exemple, si votre grille est appelée "Grid" alors le fichier XML est appelé `/Grid_spring.xml`. Ce fichier devrait se trouver dans le chemin de classe ou dans un répertoire `META-INF` qui correspond au chemin de classe. Si ce fichier est trouvé, alors eXtreme Scale construit un ApplicationContext à l'aide de ce fichier et des beans de construction provenant de cette classe de beans.

Classe de bean Spring personnalisée

WebSphere eXtreme Scale fournit également une interface de programme d'application ObjectGridSpringFactory pour enregistrer une instance de fabrique de beans Spring pour une utilisation pour un ObjectGrid spécifiquement nommé. Cette interface de programme d'application enregistre une instance de BeanFactory dans eXtreme Scale à l'aide de la méthode statique suivante :

```
void registerSpringBeanFactoryAdapter(String objectGridName, Object
springBeanFactory)
```

Prise en charge de l'espace de noms

Depuis la version 2.0, Spring dispose d'un mécanisme pour les extensions basées sur le schéma au format XML Spring de base pour la définition et la configuration des beans. ObjectGrid utilise cette nouvelle fonction pour définir et configurer les beans ObjectGrid. Avec l'extension de schéma Spring XML, une partie des implémentations des plug-in eXtreme Scale et de certains beans ObjectGrid sont prédéfinies dans l'espace de noms "objectgrid". Lors de l'écriture des fichiers de configuration Spring, il n'est pas nécessaire de spécifier le nom de classe complet des implémentations pré-intégrées. Vous pouvez plutôt référencer les beans prédéfinis.

De plus, si vous définissez les attributs des beans dans le schéma XML, cela diminue le risque de fournir un nom d'attribut erroné. La validation XML basée sur le schéma XML peut détecter ce type d'erreurs plus tôt lors du cycle de développement.

Ces beans définis dans les extensions de schéma XML sont :

- transactionManager

- registre
- serveur
- catalogue
- catalogServerProperties
- conteneur
- JPALoader
- JPATxCallback
- JPAEntityLoader
- LRUEvictor
- LFUEvictor
- HashIndex

Ces beans sont définis dans le schéma XML objectgrid.xsd XML. Ce fichier XSD est envoyé en tant que fichier `com/ibm/ws/objectgrid/spring/namespace/objectgrid.xsd` dans le fichier `ogspring.jar`. Pour obtenir une description détaillée du fichier XSD et des beans qui y sont définis, voir les informations sur le fichier descripteur Spring dans le *Guide d'administration*.

Reprenons l'exemple utilisant JPATxCallback de la section précédente. Dans la section précédente, le bean JPATxCallback est configuré comme suit :

```
<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>
```

Si elle fait appel à cette fonction d'espace de noms, la configuration Spring XML peut s'écrire comme suit :

```
<objectgrid:JPATxCallback id="jpaTxCallback" persistenceUnitName="employeeEMPU"
 jpaPropertyFactory="jpaPropFactory" />

<bean id="jpaPropFactory" class="com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl"
 scope="shard">
</bean>
```

Notez ici que plutôt que de spécifier la classe "com.ibm.websphere.objectgrid.jpa.JPATxCallback" comme dans l'exemple précédent, nous utilisons directement le bean prédéfini "objectgrid:JPATxCallback". Comme vous pouvez le voir, cette configuration est moins détaillée et facilite la détection d'erreurs.

Démarrage d'un serveur conteneur avec des beans d'extension Spring

Cet exemple illustre comment démarrer un serveur ObjectGrid à l'aide de beans d'extension gérés par ObjectGrid Spring et de la prise en charge des espaces de noms.

Fichier XML ObjectGrid

Tout d'abord, définissez un fichier ObjectGrid XML très simple qui contient un ObjectGrid "Grid" et une mappe "Test". Le fichier ObjectGrid est doté d'un plug-in ObjectGridEventListener appelé "partitionListener" et la mappe "Test" est dotée d'un expulseur appelé "testLRUEvictor". Notez que les plug-in ObjectGridEventListener et de l'expulseur sont configurés à l'aide de Spring, car leurs noms contiennent "{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>
```

Fichier XML de déploiement ObjectGrid

Maintenant, créez un fichier de déploiement XML ObjectGrid simple, comme suit. Le fichier ObjectGrid est divisé en 5 partitions et aucune réplique n'est requise.

```
<?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>
```

Fichier XML Spring ObjectGrid

Maintenant, nous allons utiliser les beans d'extension gérés pas ObjectGrid Spring et les fonctions de prise en charge d'espaces de noms pour configurer les beans ObjectGrid. Le fichier XML Spring est nommé "Grid_spring.xml". Comme vous pouvez le voir, deux schémas sont inclus dans le fichier XML : spring-beans-2.0.xsd permet d'utiliser les beans gérés par Spring et objectgrid.xsd permet d'utiliser les beans prédéfinis dans l'espace de noms 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>
```

Dans ce fichier Spring XML se trouvent 6 beans définis :

1. *objectgrid:register* : enregistre la classe de bean par défaut pour l'ObjectGrid "Grid".
2. *objectgrid:server* : définit un serveur ObjectGrid nommé "server". Ce serveur fournit également un service de catalogue, car un bean objectgrid:catalog y est imbriqué.
3. *objectgrid:catalog* : définit un point de contact de service de catalogue ObjectGrid, qui est défini sur "localhost:2809".
4. *objectgrid:container* : définit un conteneur ObjectGrid avec le fichier objectgrid XML et le fichier de déploiement XML spécifiés que nous avons évoqués précédemment. La propriété de serveur spécifie dans quel serveur ce conteneur est hébergé.
5. *objectgrid:LRUEvictor* : définit un expulseur LRUEvictor avec le nombre de files d'attente LRU à utiliser pour le définir sur 31.
6. *bean partitionListener* : définit un plug-in ShardListener. Vous devez fournir une implémentation pour ce plug-in de manière à ce qu'il ne puisse pas utiliser les beans prédéfinis. De plus, la portée du bean est définie pour "shard", ce qui signifie qu'il y a une seule instance de ShardListener par fragment ObjectGrid.

Démarrage du serveur

Le fragment ci-dessous démarre le serveur ObjectGrid, lequel héberge à la fois les services de conteneur et de catalogue. Comme nous pouvons le voir, la seule méthode à appeler pour démarrer le serveur est d'obtenir un "conteneur" de bean de la fabrique de beans. Cela simplifie la complexité de la programmation en déplaçant la plupart de la logique dans la configuration 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();
    }
}
```

Configuration du service de données REST

En suivant les liens ci-après, vous trouverez des informations sur l'administration du service de données REST. Reportez-vous également aux explications sur l'interface de programme d'application concernant le bean géré RestService.

Fichier de propriétés du service de données REST

Pour configurer le service de données REST, éditez le fichier de propriétés REST et définissez le schéma d'entité requis pour une grille WebSphere eXtreme Scale.

Le fichier de propriétés du service de données REST est le fichier de configuration principal du service de données REST d'eXtreme Scale. Ce fichier est un fichier classique de propriétés Java avec des paires clé-valeur. Par défaut, l'environnement d'exécution du service de données REST recherche dans le chemin d'accès aux classes un fichier `wxsRestService.properties` correctement nommé. Ce fichier peut également être défini de manière explicite à l'aide de la propriété système suivante : `wxs.restservice.props`.

```
-Dwxs.restservice.props=/usr/configs/dataservice.properties
```

Lorsque le service de données REST est chargé, le fichier de propriétés utilisé est affiché dans les fichiers journaux :

```
CWOBJ4004I: Les fichiers de propriétés du service de données REST de
WebSphere eXtreme Scale ont été chargés : [/usr/configs/
RestService.properties]
```

Le fichier de propriétés du service de données REST prend en charge les propriétés suivantes :

Tableau 11. Propriétés du service de données REST

Propriété	Description

catalogServiceEndPoints	La liste requise, délimitée par des virgules, des hôtes et des ports d'un domaine de services de catalogue au format <hôte:port>. Cette liste est facultative si WebSphere Application Server est intégré à eXtreme Scale pour l'hébergement du service de données REST. Voir dans la documentation de WebSphere eXtreme Scale les explications détaillées sur la manière de configurer et de démarrer un service de catalogue.
	
	`catalogServiceEndPoints=`
	
	`server1:2809,server2:2809`

objectGridNames	Les noms requis des ObjectGrid à exposer au service REST. Au moins un nom ObjectGrid est requis. Séparez par des virgules les noms d'ObjectGrids multiples : `ECommerceGrid,InventoryGrid`
objectGridClientXML	Le nom facultatif du fichier XML de remplacement ObjectGrid par les clients. Le nom spécifié ici est chargé à partir du chemin d'accès aux classes. Par défaut : /META-INF/objectGridClient.xml. Voir dans la documentation de WebSphere eXtreme Scale les explications détaillées sur la manière de configurer un client eXtreme Scale.

Tableau 11. Propriétés du service de données REST (suite)

Propriété	Description
objectGridNames	Les noms requis des ObjectGrids à exposer au service REST. Au moins un nom d'ObjectGrid est requis. Séparez par des virgules les noms d'ObjectGrids multiples : `ECommerceGrid,InventoryGrid`
objectGridClientXML	Le nom facultatif du fichier XML de remplacement ObjectGrid par les clients. Le nom spécifié ici est chargé à partir du chemin d'accès aux classes. Par défaut : /META-INF/objectGridClient.xml. Voir dans la documentation de WebSphere eXtreme Scale les explications détaillées sur la manière de configurer un client eXtreme Scale.
ogClientPropertyFile	Le nom facultatif du fichier de propriétés du client ObjectGrid. Ce fichier contient les propriétés de sécurité qui sont requises pour l'activation de la sécurité dans le client ObjectGrid. Si l'attribut securityEnabled est défini dans le fichier de propriétés, la sécurité sera activée dans le client ObjectGrid utilisé par le service REST. La propriété credentialGeneratorProps doit elle aussi être définie dans le fichier des propriétés, avec une valeur au format "user:pass" ou {xor_encoded user:pass}

Tableau 11. Propriétés du service de données REST (suite)

Propriété	Description
loginType	Le type d'authentification utilisé par le service REST lorsque la sécurité du client ObjectGrid est activée. Si cette sécurité n'est pas activée, cette propriété est ignorée. Si la sécurité du client ObjectGrid est activée et que loginType a la valeur 'basic', le service REST : • pour les opérations ObjectGrid lors de l'initialisation du service, utilisera les données d'identification spécifiées dans la propriété credentialGeneratorProps du fichier de propriétés du client ObjectGrid • pour les opérations de session ObjectGrid par demande, utilisera l'authentification Basic HTTP Si la sécurité du client ObjectGrid est activée et que loginType a la valeur 'none', le service REST : • pour les opérations ObjectGrid lors de l'initialisation du service, utilisera les données d'identification spécifiées dans la propriété credentialGeneratorProps du fichier de propriétés du client ObjectGrid • pour les opérations de session ObjectGrid par demande, utilisera les données d'identification spécifiées dans la propriété credentialGeneratorProps du fichier des propriétés du client ObjectGrid
traceFile	Nom facultatif du fichier dans lequel la sortie de la trace sera redirigée. La valeur par défaut est logs/trace.log.
traceSpec	La spécification facultative de trace que le serveur d'exécution d'eXtreme Scale doit utiliser au départ. La valeur par défaut est *=all=disabled. Pour suivre la trace de l'intégralité du service de données REST, utilisez : ObjectGridRest*=all=enabled
verboseOutput	Si la valeur est true, les clients du service de données REST reçoivent des informations de diagnostic supplémentaires en cas d'incidents. La valeur par défaut est false. Cette valeur facultative doit être false pour les environnements de production car des informations confidentielles peuvent être révélées.
maxResultsPerCollection	Le nombre maximal de résultats retournés dans une requête (facultatif). La valeur par défaut est unlimited et, pour être valide, la valeur doit être un entier positif.

Tableau 11. Propriétés du service de données REST (suite)

Propriété	Description
wxsRestAccessRightsFile	Le nom facultatif du fichier de propriétés des droits d'accès au service REST d'eXtreme Scale, qui spécifie les droits d'accès pour les opérations du service et les entités ObjectGrid. Si cette propriété est spécifiée, le service REST essaiera de charger le fichier à partir du chemin spécifié ; sinon, il essaiera de le charge à partir de son chemin d'accès aux classes.

Configuration de WebSphere eXtreme Scale

Le service de données REST d'eXtreme Scale interagit avec eXtreme Scale à l'aide de l'API EntityManager. Un schéma d'entité est défini pour une grille eXtreme Scale et les métadonnées des entités sont automatiquement utilisées par le service de données REST. Voir dans Définir un schéma d'entité les explications détaillées sur la configuration d'un schéma d'entité.

Vous pouvez, par exemple, définir de la manière suivante dans une grille eXtreme Scale une entité représentant une personne :

```
@Entity
public class Person {
  @Id String taxId;
  String firstName;
  String lastName;
}
```

Conseil : Les annotations utilisées ici se trouvent dans le package `com.ibm.websphere.projector.annotations`.

Le service REST crée automatiquement un document EDMX (Entity Data Model for Data Services) ADO.NET, accessible par l'URI $metadata :

```
http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/$metadata
```

Une fois que la grille eXtreme Scale est configurée et opérationnelle, un client eXtreme Scale doit être configuré et packagé. Pour des explications détaillées sur la configuration du package client de service de données REST d'eXtreme Scale, voir dans «Installation du service de données REST», à la page 303 les explications sur le packaging et le déploiement.

Modèle d'entité

Les entités WebSphere eXtreme Scale sont modélisées à l'aide des annotations d'entité ou d'un fichier descripteur de métadonnées d'entité. Pour savoir comment configurer un schéma d'entité eXtreme Scale, voir dans le *Guide de programmation* les explications sur la définition de schéma d'entité. Le service REST d'eXtreme Scale utilise les métadonnées d'entité pour créer automatiquement un modèle EDMX pour le service de données.

Cette version du service de données REST de WebSphere eXtreme Scale comporte les restrictions de schéma suivantes :

- Lors de la définition d'entités dans une grille partitionnée, toutes les entités doivent posséder une association directe ou indirecte à valeur unique avec

l'entité racine (association de clé). L'environnement d'exécution du client du service de données WCF doit pouvoir accéder à toutes les entités directement via son adresse canonique. Par conséquent, la clé de l'entité racine utilisée pour le routage des partitions (racine du schéma) doit faire partie de la clé dans l'entité enfant.

Par exemple :

```
@Entity(schemaRoot=true)
public class Person {
    @Id String taxId;
    String firstName;
    String lastName;
    @OneToMany(mappedBy="person")
    List<Address> addresses;

}

@Entity
public class Address {
    @Id int addrId;
    @Id @ManyToOne Person person;
    String street;
}
```

- Les associations unidirectionnelles et bidirectionnelles sont prises en charge. Toutefois, les associations risquent de ne pas toujours fonctionner à partir d'un client Microsoft WCF Data Services car elles ne sont parcourues que dans un sens et la spécification Microsoft requiert que toutes les associations soient bidirectionnelles.
- Les contraintes référentielles ne sont pas prises en charge. L'environnement d'exécution d'eXtreme Scale ne valide pas les clés entre les entités. Les associations entre les entités doivent être gérées par le client.
- Les types complexes ne sont pas pris en charge. L'API EntityManager d'eXtreme Scale ne prend pas en charge les attributs incorporables. Tous les attributs doivent être des attributs de type simple (voir les attributs de type simple répertoriés ci-après). Les attributs non simples sont traités comme un objet binaire du point de vue du client.
- L'héritage des entités n'est pas pris en charge. L'API EntityManager d'eXtreme Scale ne prend pas en charge l'héritage.
- Les ressources de support et les liens de support ne sont pas pris en charge. L'attribut HasStream du type d'entité dans le document CSDL (Conceptual Schema Definition Language) des services de données n'est jamais utilisé.

Mappage entre les types de données EDM et les types de données Java

Le protocole OData définit la liste ci-après des types de modèle de données d'entité (EDM) et son système de type abstrait. Les rubriques qui suivent décrivent comment l'adaptateur REST d'eXtreme Scale choisit le type EDM en fonction du type de base défini dans l'entité. Pour des détails sur les types EDM, voir MSDN Library: Abstract Type System.

Les types EDM suivants sont disponibles dans les services de données WCF :

- Edm.Binary
- Edm.Boolean
- Edm.Byte
- Edm.DateTime
- Edm.Time

- Edm.Decimal
- Edm.Double
- Edm.Single
- Edm.Float
- Edm.Guid *
- Edm.Int16
- Edm.Int32
- Edm.Int64
- Edm.SByte
- Edm.String

Le type EDM Edm.Guid n'est pas pris en charge par le service de données REST d'eXtreme Scale.

Mappage des types Java avec les types EDM

Le service de données REST d'eXtreme Scale convertira automatiquement les types d'entité de base en types EDM. Le mappage des types peut être consulté en affichant le document des métadonnées EDMX (Entity Data Model Extensions) à l'aide de l'URI $metadata. Le type EDM est utilisé par les clients pour lire et écrire les données dans le service de données REST.

Tableau 12. Types Java mappés à des types EDM. Le tableau montre le mappage vers un type de données EDM du type Java défini pour une entité. Lors de l'extraction de données à l'aide d'une requête, les données seront représentées avec ces types :

Type Java	**Type EDM**
boolean java.lang.Boolean	Edm.Boolean
byte java.lang.Byte	Edm.SByte
short java.lang.Short	Edm.Int16
int java.lang.Integer	Edm.Int32
long java.lang.Long	Edm.Int64
float java.lang.Float	Edm.Single
double java.lang.Double	Edm.Double
java.math.BigDecimal	Edm.Decimal
java.math.BigInteger	java.math.BigInteger
java.lang.String	Edm.String
char	char
java.lang.Character	java.lang.Character
Char[]	Char[]
java.lang.Character[]	java.lang.Character[]
java.util.Calendar	Edm.DateTime
java.util.Date	java.util.Date
java.sql.Date	java.sql.Date
java.sql.Timestamp	java.sql.Timestamp
java.sql.Time	java.sql.Time
Autres types	Edm.Binary

Mappage des types EDM vers les types Java

Pour les demandes Update et Insert, la charge spécifie les données à actualiser ou à insérer dans le service de données REST d'eXtreme Scale. Le service peut automatiquement convertir les types de données compatibles vers les types de données définis dans le document EDMX. Le service de données REST convertit les représentations de chaîne codées en XML de la valeur dans le type approprié à l'aide de la procédure en deux étapes suivante :

1. Une vérification du type est effectuée pour s'assurer que le type EDM est compatible avec le type Java. Un type EDM est compatible avec un type Java si les données prises en charge par le type EDM sont un sous-ensemble des données prises en charge par le type Java. Par exemple, le type Edm.int32 est compatible avec un type Java long, mais le type Edm.int32 n'est pas compatible avec un type Java short.
2. Un objet cible de type Java est créé pour représenter la valeur de la chaîne dans la charge.

Tableau 13. Types EDM compatibles avec les types Java

Type EDM	**Type Java**
Edm.Boolean	boolean java.lang.Boolean
Edm.SByte	byte java.lang.Byte short java.lang.Short int java.lang.Integer long java.lang.Long float java.lang.Float double java.lang.Double java.math.BigDecimal java.math.BigInteger char java.lang.Character

Tableau 13. Types EDM compatibles avec les types Java (suite)

Type EDM	Type Java
Edm.Byte, Edm.Int16	short java.lang.Short entier java.lang.Integer long java.lang.Long float java.lang.Float double java.lang.Double java.math.BigDecimal java.math.BigInteger char java.lang.Character
Edm.Int32	entier java.lang.Integer long java.lang.Long float java.lang.Float double java.lang.Double java.math.BigDecimal java.math.BigInteger
Edm.Int64	long java.lang.Long double java.lang.Double java.math.BigDecimal java.math.BigInteger

Tableau 13. Types EDM compatibles avec les types Java (suite)

Type EDM	Type Java
Edm.Double	double java.lang.Double java.math.BigDecimal
Edm.Decimal	double java.lang.Double java.math.BigDecimal java.math.BigInteger
Edm.Single	float java.lang.Float double java.lang.Double java.math.BigDecimal
Edm.String	java.lang.String char java.lang.Character Char[] java.lang.Character[] java.math.BigDecimal java.math.BigInteger
Edm.DateTime	java.util.Calendar java.util.Date java.sql.Date java.sql.Time java.sql.Timestamp
Edm.Time	java.sql.Time java.sql.Timestamp

Mappage des types temporels

Java inclut cinq types temporels permettant de stocker la date et/ou l'heure : java.util.Date, java.sql.Date, java.sql.Time, java.sql.Timestamp et java.util.Calendar. Tous ces types sont exprimés dans le modèle de données d'entité au format Edm.DateTime. Le service REST d'eXtreme Scale convertit et normalise automatiquement les données en fonction du type Java. Cette rubrique décrit plusieurs problèmes dont les développeurs doivent être conscients lorsqu'ils utilisent un type temporel.

Décalages horaires

Dans WCF Data Services, les descriptions des valeurs temporelles dans le type Edm.DateTime sont toujours exprimées à l'aide de la norme UTC (temps universel coordonné), qui correspond au nom international du temps moyen de Greenwich (GMT). Le temps universel coordonné correspond à l'heure mesurée à la longitude de zéro degrés (le point d'origine UTC). L'heure d'été ne s'applique pas à l'heure UTC.

Conversion entre les types d'entité et les types EDM

Lorsqu'un client envoie une demande au service de données REST, la date et l'heure sont représentées comme une heure GMT, comme dans l'exemple suivant :

```
"2000-02-29T21:30:30.654123456"
```

Le service de données REST génère ensuite l'instance Java de type temporel appropriée et l'insère dans l'entité, dans la grille.

Lorsqu'un client demande au service de données REST d'eXtreme Scale une propriété qui est un type Java temporel, la valeur est toujours normalisée en valeur GMT. Par exemple, si une entité java.util.Date est construite comme suit :

```
Calendar c = Calendar.getInstance();
c.clear();
c.set(2000, 1, 29, 21, 30, 30);
Date d = c.getTime();
```

La date et l'heure sont représentées à l'aide du fuseau horaire par défaut du processus Java car Calendar.getInstance() crée un objet Calendar avec le fuseau horaire local. Si, le fuseau horaire local est CST, la date, lorsqu'elle est extraite du service de données REST correspond à la représentation GMT de l'heure :
`"2000-03-01T03:30:30"`

Normalisation de java.sql.Date

Une entité eXtreme Scale peut définir un attribut avec le type Java java.sql.Date. Ce type de données n'inclut pas l'heure et est normalisé par le service de données REST. Cela signifie que l'environnement d'exécution d'eXtreme Scale ne stocke pas les heures, minutes, secondes ou millisecondes dans l'attribut java.sql.Date. Quel que soit le décalage horaire, la date est toujours représentée comme une date locale.

Par exemple, si le client met à jour une propriété java.sql.Date avec la valeur “2009-01-01T03:00:00”, le service de données REST, qui se trouve dans le fuseau horaire CST (-06:00), crée simplement une instance java.sql.Date dont l'heure est “2009-01-01T00:00:00” (heure CST locale). Aucun décalage horaire n'est appliqué pour créer la valeur java.sql.Date. Lorsque le client du service REST extrait la valeur de cet attribut, cette dernière est affichée sous la forme “2009-01-01T00:00:00Z”. Si un décalage horaire était appliqué, la valeur serait affichée avec la date “2008-12-31”, ce qui ne serait pas correct.

Normalisation de java.sql.Time

Comme pour java.sql.Date, les valeurs java.sql.Time sont normalisées et n'incluent pas la date. Cela signifie que l'environnement d'exécution d'eXtreme Scale ne

stocke pas l'année, le mois ou le jour. L'heure est stockée en heure GMT à partir du 1er janvier 1970, ce qui est cohérent avec l'implémentation de java.sql.Time.

Par exemple, si le client met à jour une propriété java.sql.Time avec la valeur "2009-01-01T03:00:00", le service de données REST crée une instance java.sql.Time avec une valeur de 3*60*60*1000 millisecondes, soit 3 heures. Lorsque le service REST extrait la valeur, cette dernière est affichée sous la forme "1970-01-01:03:00:00Z".

Associations

Les associations définissent la relation entre deux entités homologues. Le service REST d'eXtreme Scale reflète les associations modélisées avec des entités définies à l'aide des entités annotées d'eXtreme Scale ou des entités définies à l'aide d'un fichier XML de descripteur d'entité.

Maintenance de l'association

Le service de données REST d'eXtreme Scale ne prend pas en charge les contraintes d'intégrité référentielles. Le client doit s'assurer que les références sont mises à jour lorsque des entités sont supprimées ou ajoutées. Si une entité cible d'une association est supprimée de la grille, mais que le lien entre l'entité source et l'entité cible n'est pas supprimé, le lien est rompu. Le service de données REST d'eXtreme Scale et l'API EntityManager tolèrent les liens rompus et les consignent comme avertissements CWPRJ1022W. Les associations rompues sont simplement supprimées de la charge de la demande.

Utilisez une demande par lots pour regrouper les mises à jour d'association dans une même transaction afin d'éviter les liens rompus. Voir la section pour des explications détaillées sur les demandes par lots.

L'élément ReferentialConstraint du modèle de données d'entité ADO.NET n'est pas utilisé par le service de données REST d'eXtreme Scale.

Multiplicité des associations

Les entités peuvent avoir des associations à plusieurs valeurs ou des associations à valeur unique. Les associations à plusieurs valeurs ou collections sont des associations one-to-many ou many-to-many. Les associations à valeur unique sont des associations one-to-one ou many-to-one.

Dans une grille partitionnée, toutes les entités doivent posséder un chemin d'association de clés à valeur unique à une entité racine. Une autre section de cette rubrique explique comment définir une association de clés. L'entité racine étant utilisée pour partitionner l'entité, les associations many-to-many ne sont pas autorisées pour les grilles partitionnées. Pour un exemple de comment modéliser un schéma d'entité relationnelle pour une grille partitionnée, voir Evolutivité du modèle de données dans eXtreme Scale.

L'exemple suivant décrit comment les types d'association de l'API EntityManager, modélisés à l'aide de classes Java annotées, sont mappés au modèle de données d'entité ADO.NET :

```
@Entity
public class Customer {
    @Id String customerId;
    @OneToOne TaxInfo taxInfo;
```

```
    @ManyToOne Address homeAddress;
    @OneToMany Collection<Order> orders;
    @ManyToMany Collection<SalesPerson> salespersons;
}

<Association Name="Customer_TaxInfo">
    <End Type="Model1.Customer" Role="Customer" Multiplicity="1" />
    <End Type="Model1.TaxInfo " Role="TaxInfo" Multiplicity="1" />
</Association>
<Association Name="Customer_Address">
    <End Type="Model1.Customer" Role="Customer" Multiplicity="1" />
    <End Type="Model1.Address" Role="TaxInfo" Multiplicity="*" />
</Association>
<Association Name="Customer_Order">
    <End Type="Model1.Customer" Role="Customer" Multiplicity="*" />
    <End Type="Model1.Order" Role="TaxInfo" Multiplicity="1" />
</Association>
<Association Name="Customer_SalesPerson">
    <End Type="Model1.Customer" Role="Customer" Multiplicity="*" />
    <End Type="Model1.SalesPerson" Role="TaxInfo" Multiplicity="*" />
</Association>
```

Associations unidirectionnelles et bidirectionnelles

Les associations d'entités peuvent être unidirectionnelles ou bidirectionnelles. En spécifiant l'attribut "mappedBy" sur l'annotation @OneToOne, @OneToMany ou @ManyToMany ou l'attribut "mapped-by" sur la balise d'attribut XML one-to-one, one-to-many ou many-to-many, l'entité devient bidirectionnelle. Le protocole OData requiert actuellement que toutes les entités soient bidirectionnelles, afin de permettre aux clients de générer des chemins de navigation dans les deux sens. L'API EntityManager d'eXtreme Scale autorise la modélisation d'associations unidirectionnelles qui permettent d'économiser la mémoire et de simplifier la maintenance des associations. Si une association unidirectionnelle est utilisée, le client des services de données REST ne doit naviguer dans l'association qu'à l'aide de l'association définie.

Par exemple : Si une association unidirectionnelle many-to-one est définie entre Address et Country, l'URI n'est pas autorisé :

```
/restservice/CustomerGrid/Country('USA')/addresses
```

Associations clés

Des associations à valeur unique (one-to-one et many-to-one) peuvent également être incluses comme une clé d'entités ou partie intégrante de cette clé. Il s'agit d'une association clé.

Les associations clés sont requises si une grille partitionnée est utilisée. L'association clé doit être définie pour toutes les entités enfant d'un schéma d'entités partitionné. Le protocole OData requiert que toutes les entités soient directement adressables. Cela signifie que la clé de l'entité enfant doit inclure la clé utilisée pour le partitionnement.

Dans l'exemple ci-après, Customer fait l'objet d'une association one-to-many avec Order. L'entité Customer représente l'entité racine et l'attribut customerId est utilisé pour partitionner l'entité. La commande a inclus le client dans son identité :

```
@Entity(schemaRoot="true")
public class Customer {
    @Id String customerId;
    @OneToMany(mappedBy="customer") Order orders
```

```
}

@Entity
public class Order {
    @Id int orderId;
    @Id @ManyToOne Customer customer;
    java.util.Date orderDate;
}
```

Lorsque le service de données REST génère le document EDMX pour ce modèle, les champs de la clé Customer sont automatiquement inclus dans l'entité Order :

```
<EntityType Name="Order">
<Key>
    <PropertyRef Name="orderId"/>
    <PropertyRef Name="customer_customerId"/>
</Key>

<Property Name="orderId" Type="Edm.Int64" Nullable="false"/>
<Property Name="customer_customerId" Type="Edm.String"
    Nullable="false"/>
<Property Name="orderDate" Type="Edm.DateTime" Nullable="true"/>
<NavigationProperty Name="customer"
    Relationship="NorthwindGridModel.Customer_orders"
    FromRole="Order" ToRole="Customer"/>

<NavigationProperty Name="orderDetails"
    Relationship="NorthwindGridModel.Order_orderDetails"
    FromRole="Order" ToRole="OrderDetail"/>
</EntityType>
```

Lorsqu'une entité est créée, la clé ne doit jamais être modifiée. Cela signifie que l'association clé entre une entité enfant et son parent doit être modifiée ; l'entité enfant doit être supprimée et recréée avec un autre parent. Dans une grille partitionnée, cela nécessite deux ensembles de modifications par lots car le transfert risque d'impliquer plusieurs partitions.

Opérations en cascade

L'API EntityManager accepte les règles en cascade flexibles. Les associations peuvent être marquées de sorte à réaliser en cascade des opérations de persistance, de suppression, d'invalidation ou de fusion. De telles opérations de cascade peuvent se produire des deux côtés d'une association bidirectionnelle.

Le protocole OData n'autorise les opérations de suppression en cascade que d'un côté de l'association. L'annotation CascadeType.REMOVE ou l'attribut XML cascade-remove ne peut pas être défini des deux côtés d'une association bidirectionnelle one-to-one ou du côté "many" d'une association one-to-many. L'exemple suivant illustre une association bidirectionnelle Cascade.REMOVE valide :

```
@Entity(schemaRoot="true")
public class Customer {
    @Id String customerId;
    @OneToMany(mappedBy="customer", cascade=CascadeType.REMOVE)
    Order orders
}

@Entity
public class Order {
    @Id int orderId;
    @Id @ManyToOne Customer customer;
    java.util.Date orderDate;
}
```

L'association EDMX résultante se présente comme suit :

```
<Association Name="Customer_orders">
    <End Type="NorthwindGridModel.Customer" Role="Customer"
        Multiplicity="1">
        <OnDelete Action="Cascade"/>
    </End>
    <End Type="NorthwindGridModel.Order" Role="Order"
        Multiplicity="*"/>
</Association>
```

Administration du service de données REST

Pourquoi et quand exécuter cette tâche

En suivant les liens ci-après, vous trouverez des informations sur l'administration du service de données REST. Voir également les informations concernant le bean géré RestService.

Installation du service de données REST

Nous allons expliquer comment installer sur un serveur Web le service de données REST d'WebSphere eXtreme Scale.

Avant de commencer

Configuration logicielle

Le service de données REST d'eXtreme Scale est une application Web Java qui peut être déployée sur tout serveur d'applications prenant en charge la spécification de servlet Java version 2.3 et un environnement d'exécution Java version 5 ou plus récente.

Les logiciels suivants sont requis :

- Java Standard Edition 5 ou version ultérieure

  Restriction : Bien que eXtreme Scale prenne en charge Java Standard Edition version 1.4 ou ultérieure, le service de données REST requiert Java Standard Edition version 5 ou ultérieure.
- le conteneur de servlets Web version 2.3 ou ultérieure, qui inclut l'un des éléments suivants :
  - le serveur d'applications WebSphere version 6.1.0.25 ou ultérieure
  - le serveur d'applications WebSphere version 7.0.0.5 ou ultérieure
  - WebSphere Community Edition version 2.1.1.3 ou ultérieure
  - Apache Tomcat version 5.5 ou ultérieure
- eXtreme Scale version 7.1 ou ultérieure (y compris la version d'évaluation)

Pourquoi et quand exécuter cette tâche

Le service de données REST d'eXtreme Scale comprend un seul fichier WAR, `wxsrestservice.war`. Le fichier `wxsrestservice.war` comporte un seul servlet qui fait office de passerelle entre vos applications de client WCF Data Services ou tout autre client REST HTTP et une grille eXtreme Scale.

Le service de données REST inclut un échantillon permettant de créer rapidement une grille eXtreme Scale et d'interagir avec cette grille à l'aide d'un client eXtreme Scale ou du service de données REST. Voir Exemple et tutoriel sur les services de données REST.

Lors de l'installation de eXtreme Scale 7.1 ou de l'extraction de la version d'évaluation d'eXtreme Scale version 7.1, les répertoires et fichiers suivants sont inclus :

- `base_servicerest/lib`

  Le répertoire lib contient ces fichiers :

  – `wxsrestservice.ear` – L'archive d'application d'entreprise de service de données REST à utiliser avec le serveur d'application WebSphere et le serveur d'application CE WebSphere.

  – `wxsrestservice.war` – Le module Web de service de données REST à utiliser avec Apache Tomcat.

  Le fichier `wxsrestservice.ear` inclut le fichier `wxsrestservice.war` et tous deux sont étroitement couplés à l'environnement d'exécution WebSphere eXtreme Scale. En cas de mise à niveau d'eXtreme Scale vers une nouvelle version ou si un groupe de correctifs est appliqué, les fichiers `wxsrestservice.war` file ou `wxsrestservice.ear` devront être mis à niveau manuellement vers la version installée dans ce répertoire.

- `base_servicerest/gettingstarted`

  Le répertoire `gettingstarted` contient un échantillon simple indiquant comment utiliser le service de données REST d'eXtreme Scale avec une grille eXtreme Scale.

Procédure

Packagez et déployez le service de données REST.
Le service de données REST a été conçu en tant que module WAR autonome. Pour configurer le service de données REST, vous devez commencer par packager dans un fichier JAR ou dans un répertoire la configuration du service de données REST et les éventuels fichiers de configuration d'eXtreme Scale. Ce package d'application est ensuite référencé par l'environnement d'exécution du serveur de conteneur Web. La figure suivante illustre les fichiers utilisés par le service de données REST d'eXtreme Scale.

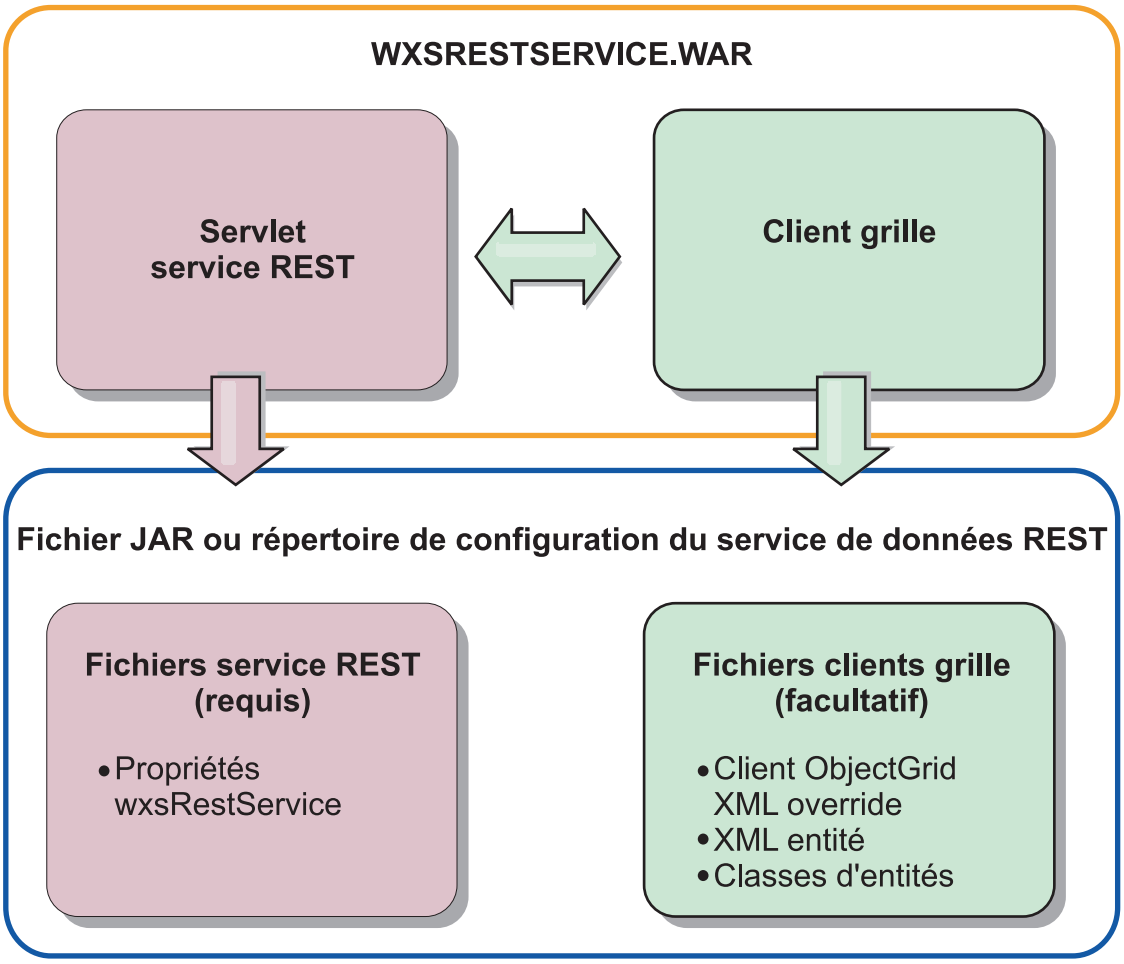


Figure 18. Fichiers du service de données REST d'WebSphere eXtreme Scale

Le fichier JAR de configuration du service REST ou le répertoire doit contenir le fichier suivant :
`wxsRestService.properties` : le fichier `wxsRestService.properties` comprend les options de configuration du service de données REST : points de contact du service de catalogue, noms d'ObjectGrid à exposer, options de suivi, etc. Voir «Fichier de propriétés du service de données REST», à la page 290.
Les fichiers suivants du client ObjectGrid sont facultatifs :

- `META-INF/objectGridClient.xml` : le fichier XML de remplacements par le client sert à se connecter à la grille eXtreme Scale distante. Par défaut, ce fichier n'est pas requis. En son absence, le service REST utilise la configuration du serveur en désactivant le cache proche.

  Le nom du fichier peut être remplacé à l'aide de la propriété de configuration du service de données REST objectGridClientXML. S'il est fourni, ce fichier XML doit inclure :

  1. tous les ObjectGrids que vous souhaitez exposer au service de données REST
  2. une référence au fichier XML du descripteur d'entité associé à chaque configuration ObjectGrid
- `META-INF/`*`fichiers XML de descripteurs d'entités`* : un ou plusieurs fichiers XML de descripteurs d'entités ne sont requis que si le client doit remplacer la définition d'entité du client. Le fichier XML du descripteur d'entité doit être utilisé avec le fichier XML du descripteur d'entité de remplacement ObjectGrid par les clients.

For details on the eXtreme Scale configuration files, see the eXtreme Scale *Guide d'administration*..

- **Classes entité** Vous pouvez utiliser des classes entité annotées ou un fichier XML descripteur d'entité pour décrire les métadonnées d'entité. Le service REST ne nécessite les classes d'entités dans le chemin d'accès aux classes que si les serveurs eXtreme Scale sont configurés avec les classes de métadonnées d'entités. Aucun fichier XML de descripteur d'entité de remplacement par le client n'est utilisé.

  Voici un exemple avec le fichier de configuration minimum requise, où les entités sont définies dans XML sur les serveurs :

  ```
  restserviceconfig.jar:
  wxsRestService.properties
  ```

  Le fichier de propriétés contient :

  ```
  catalogServiceEndPoints=localhost:2809
  objectGridNames=NorthwindGrid
  ```

  Un exemple avec une entité, des fichiers XML de remplacement et des classes entité :

  ```
  restserviceconfig.jar:
  wxsRestService.properties
  ```

  Le fichier de propriétés contient :

  ```
  catalogServiceEndPoints=localhost:2809
  objectGridNames=NorthwindGrid
  ```

  ```
  com/acme/entities/Customer.class
  META-INF/objectGridClient.xml
  ```

  Le fichier XML descripteur du client ObjectGrid contient :

  ```
  <objectGrid name="CustomerGrid" entityMetadataXMLFile="emd.xml"/>
  META-INF/emd.xml
  ```

  Le fichier XML descripteur des métadonnées d'entité contient :

  ```
  <entity class-name="com.acme.entities.Customer" name="Customer"/>
  ```

  Pour des explications détaillées sur l'API EntityManager et la configuration d'un client et d'un serveur eXtreme Scale, voir le *Guide d'administration*.

Déploiement du service de données REST sur WebSphere Application Server

Nous allons expliquer comment configurer le service de données REST d'eXtreme Scale sur WebSphere Application Server ou sur WebSphere Network Deployment version 6.1.0.25 ou plus récente. Ces instructions s'appliquent également aux déploiements où WebSphere eXtreme Scale est intégré au déploiement de WebSphere Application Server.

Avant de commencer

Votre système doit utiliser l'un des environnements ci-après pour que vous puissiez configurer et déployer le service de données REST de WebSphere eXtreme Scale.

- WebSphere Application Server avec le client autonome d'eXtreme Scale :
  - La version d'évaluation de WebSphere eXtreme Scale version 7.1. avec le service de données REST est téléchargée et extraite ou le produit WebSphere eXtreme Scale version 7.1.0.0 avec le groupe de correctifs 2 est installé dans un répertoire autonome.
  - WebSphere Application Server Version 6.1.0.25 ou 7.0.0.5 (ou ultérieure) est installé et actif.
- WebSphere Application Server intégré à WebSphere eXtreme Scale :

WebSphere eXtreme Scale version 7.1.0.0 avec le groupe de correctifs 2 est installé par dessus WebSphere Application Server version 6.1.0.25 ou 7.0 (ou plus récente).

Conseil : Le service de données REST d'eXtreme Scale requiert uniquement que le client eXtreme Scale soit installé. Il n'est pas nécessaire d'étendre le profil.

Consultez sur le Centre de documentation de WebSphere Application Server les explications sur la manière d'activer la sécurité Java 2.

Procédure

1. Configurez et démarrez une grille eXtreme Scale.
   a. Pour des explications détaillées sur la configuration d'une grille eXtreme Scale à utiliser avec le service de données REST, voir Chapitre 6, «Configuration de l'environnement de déploiement», à la page 97.
   b. Vérifiez qu'un client eXtreme Scale arrive à se connecter aux entités de la grille et à y accéder. Pour un exemple, reportez-vous à la section Mise en route de ce document.
2. Générez le répertoire ou le fichier JAR de configuration du service REST d'eXtreme Scale. Reportez-vous dans «Installation du service de données REST», à la page 303 aux explications sur le packaging et le déploiement du service REST.
3. Ajoutez le fichier JAR ou le répertoire de la configuration du service de données REST au chemin d'accès aux classes du serveur d'applications :
   a. Ouvrez la console d'administration WebSphere
   b. Accédez à **Environnement → Bibliothèques partagées**
   c. Cliquez sur **Nouveau**
   d. Ajoutez les entrées suivantes dans les zones appropriées :
      - Nom : `extremescale_rest _configuration`
      - Chemin de classes : <répertoire ou fichier JAR de configuration du service REST>
   e. Cliquez sur **OK**
   f. Sauvegardez les modifications apportées à la configuration principale
4. Si eXtreme Scale est intégré à l'installation de WebSphere Application Server, ignorez cette étape et passez au point 5. Sinon, continuez :

   Ajoutez au chemin d'accès aux classes du serveur d'applications le fichier JAR d'exécution du client WebSphere eXtreme Scale (wsogclient.jar) et le fichier JAR ou le répertoire de configuration du service de données REST :
   a. Ouvrez la console d'administration WebSphere
   b. Accédez à **Environnement → Bibliothèques partagées**
   c. Cliquez sur **Nouveau**
   d. Ajoutez les entrées suivantes dans les zones :
      - Nom : `extremescale_client_v71`
      - Chemin d'accès aux classes : `base_wxs/lib/wsogclient.jar`
   e. Cliquez sur OK
   f. Sauvegardez les modifications apportées à la configuration principale
5. Installez le fichier EAR du service de données REST, wxsrestservice.ear, dans WebSphere Application Server, à l'aide de la console d'administration de WebSphere :
   a. Ouvrez la console d'administration WebSphere

b. Accédez à Applications -> Nouvelle application
c. Accédez au fichier /lib/wxsrestservice.ear sur le système de fichiers, sélectionnez-le et cliquez sur **Suivant**.
   - Si vous utilisez WebSphere Application Server version 7.0, cliquez sur Suivant.
   - Si vous utilisez WebSphere Application Server version 6.1, entrez la valeur de racine de contexte nommée /wxsrestservice et passez à l'étape suivante.
d. Choisissez l'option d'installation détaillée et cliquez sur Suivant.
e. Dans l'écran des avertissements de sécurité de l'application, cliquez sur Continuer.
f. Choisissez les options d'installation par défaut et cliquez sur Suivant.
g. Choisissez un serveur auquel l'application sera mappée et cliquez sur Suivant.
h. Dans la page de rechargement JSP, utilisez les valeurs par défaut et cliquez sur Suivant.
i. Dans la page des bibliothèques partagées, mappez le module "wxsrestservice.war" aux bibliothèques partagées suivantes, définies dans les étapes 3 et 4 :
   - extremescale_rest _configuration
   - extremescale_client_v71

   Conseil : Cette bibliothèque partagée n'est requise que si eXtreme Scale n'est pas intégré à WebSphere Application Server.
j. Dans la page des relations de la bibliothèque partagée des mappes, utilisez les valeurs par défaut et cliquez sur Suivant.
k. Dans la page des hôtes virtuels des mappes, utilisez les valeurs par défaut et cliquez sur Suivant.
l. Dans la page des racines de contexte des mappes, spécifiez wxsrestservice comme racine de contexte.
m. Dans l'écran Récapitulatif, cliquez sur Terminer pour terminer l'installation.
n. Sauvegardez les modifications apportées à la configuration principale.

6. Démarrez wxsrestservice, l'application du service de données REST d'eXtreme Scale :
   a. Choisissez l'application.
      - Si vous utilisez WebSphere Application Server version 7.0 : Dans la console d'administration, cliquez sur **Applications → Types d'application → Applications WebSphere**
      - Si vous utilisez WebSphere Application Server version 6.1 : Dans la console d'administration, cliquez sur **ApplicationsApplications d'entreprise**.
   b. Cochez la case en regard de l'application "wxsrestservice ", puis cliquez sur **Démarrer**.
   c. Recherchez le profil du serveur d'applications dans le fichier SystemOut.log. Si le service de données REST est correctement démarré, le message suivant est affiché dans le fichier SystemOut.log du profil du serveur :

      ```
      CWOBJ4000I: Le service de données REST de WebSphere eXtreme Scale a été démarré.
      ```
7. Vérifiez que le service de données REST fonctionne : Vous trouverez le numéro de port dans le fichier SystemOut.log du répertoire des journaux du profil de

serveur d'applications en recherchant le premier port affiché pour le message dont l'identificateur est SRVE0250I. Le port par défaut est 9080.

Par exemple : http://localhost:9080/wxsrestservice/restservice/NorthwindGrid/ Résultat : Le document de service AtomPub est affiché.

Déploiement du service de données REST sur WebSphere Application Server Community Edition

Nous allons expliquer comment configurer le service de données REST d'eXtreme Scale sur WebSphere Application Server Community Edition version 2.1.1.3 ou plus récente.

Avant de commencer

- Un JRE ou JDK IBM (recommandé) ou Sun, version 5 ou ultérieure, est installé et la variable d'environnement JAVA_HOME est définie.
- Téléchargez et installez WebSphere Application Server Community Edition version 2.1.1.3 ou plus récente dans le répertoire `racine_wasce`, par exemple le répertoire `/opt/IBM/wasce`. Pour plus d'informations sur version 2.1.1 ou (autres versions), lisez les instructions d'installation.
- La version d'évaluation d'eXtreme Scale version 7.1. avec le service de données REST est téléchargée et extraite ou le produit WebSphere eXtreme Scale version 7.1.0.0 avec le groupe de correctifs 2 est installé dans un répertoire autonome.

Procédure

1. Configurez et démarrez une grille eXtreme Scale.
   a. Pour des explications détaillées sur la configuration d'une grille eXtreme Scale à utiliser avec le service de données REST, voir Chapitre 6, «Configuration de l'environnement de déploiement», à la page 97.
   b. Vérifiez qu'un client eXtreme Scale arrive à se connecter aux entités de la grille et à y accéder. Pour un exemple, voir Exemple et tutoriel sur les services de données REST.
2. Générez le répertoire ou le fichier JAR de configuration du service REST d'eXtreme Scale. Pour les détails, voir dans «Installation du service de données REST», à la page 303 les explications sur le packaging et le déploiement.
3. Démarrez le serveur WebSphere Application Server Community Edition :
   a. Pour démarrer le serveur sans la sécurité Java SE activée, exécutez la commande suivante :

      UNIX Linux `racine_wasce/bin/startup.sh`

      Windows `racine_wasce/bin/startup.bat`

   b. Pour démarrer le serveur avec la sécurité Java SE activée, effectuez les étapes suivantes : UNIX Linux
      1) Open a command-line or terminal window and run the following copy command (or copy the contents of the specified policy file into your existing policy): `cp base_servicerest/gettingstarted/wasce/geronimo.policy wasce_root/bin`
      2) Modifiez le fichier `racine_wasce/bin/setenv.sh`.
      3) Après la ligne qui contient `"WASCE_JAVA_HOME="`, ajoutez la ligne suivante : `export JAVA_OPTS="-Djava.security.manager -Djava.security.policy=geronimo.policy"`.

      Windows

1) Ouvrez une fenêtre de ligne de commande et exécutez la commande de copie suivante (ou copiez le contenu du fichier de règle spécifié dans votre règle existante):

   ```
   copy base_servicerest\gettingstarted\wasce\geronimo.policy\bin
   ```

2) Modifiez le fichier `racine_wasce/bin/setenv.bat`.
3) Après la ligne qui contient "set WASCE_JAVA_HOME=", ajoutez la ligne suivante :

   ```
   set JAVA_OPTS="-Djava.security.manager
   -Djava.security.policy=geronimo.policy"
   ```

4. Ajoutez le fichier JAR d'exécution du client ObjectGrid au référentiel WebSphere Application Server Community Edition :
   a. Ouvrez et connectez-vous à la console d'administration de WebSphere Application Server Community Edition. L'URL par défaut est : http://localhost:8080/console, l'ID utilisateur par défaut est "system" et le mot de passe est "manager".
   b. Cliquez sur le lien **Référentiel** situé dans la partie gauche de la fenêtre de la console, dans le dossier **Services**.
   c. Dans la section **Ajouter une archive au référentiel**, entrez les éléments suivants dans les zones de texte :

Tableau 14. Ajout d'une archive au référentiel

Zone de texte	Valeur
Fichier	`base_wxs/lib/ogclient.jar`
Groupe	com.ibm.websphere.xs
Artefact	ogclient
Version	7.1
Type	JAR

   d. Cliquez sur le bouton Installer

   Reportez-vous à la note technique suivante pour des détails sur les différentes manières dont les dépendances de classes et de bibliothèques peuvent être configurées : Specifying external dependencies to applications running on WebSphere Application Server Community Edition.
5. Déployez vers le serveur WebSphere Application Server Community Edition le module du service de données REST, le fichier `wxsrestservice.war`, et démarrez-le.
   a. Copiez et éditez l'exemple de fichier XML de plan de déploiement : `base_servicerest/gettingstarted/wasce/geronimo-web.xml`, afin d'inclure les dépendances de chemin au répertoire ou au fichier JAR de configuration de votre service de données REST. Voir la section pour un exemple de définition du chemin d'accès aux classes afin d'y inclure votre fichier `wxsRestService.properties` ainsi que d'autres fichiers de configuration et classes de métadonnées.
   b. Ouvrez et connectez-vous à la console d'administration de WebSphere Application Server Community Edition.

      Conseil : L'URL par défaut est : http://localhost:8080/console. L'ID utilisateur et le mot de passe par défaut sont respectivement "system" et "manager".
   c. Cliquez sur le lien **Déployer nouveau** situé dans la partie gauche de la fenêtre de la console.

d. Entrez les valeurs suivantes dans les zones de texte de la page **Installer de nouvelles applications** :

Tableau 15. Installer de nouvelles applications

Zone de texte	Valeur
Archive	base_servicerest/lib/wxsrestservice.war
Plan	base_servicerest/gettingstarted/wasce/geronimo-web.xml

Conseil : Utilisez le chemin du fichier `geronimo-web.xml` que vous avez copié et édité au point 3.

e. Cliquez sur le bouton Installer. La page de la console indique alors que l'application a été installée et démarrée.

f. Recherchez le message ci-après sur la console ou dans le journal de sortie système de WebSphere Application Server Community Edition pour vérifier que le service de données REST a bien démarré :

```
CWOBJ4000I: Le service de données REST de WebSphere eXtreme Scale a été démarré.
```

6. Démarrez le serveur WebSphere Application Server Community Edition en exécutant la commande suivante :
   - UNIX Linux `racine_wasce/bin/startup.sh`
   - Windows `racine_wasce/bin/startup.bat`
7. Installez sur le serveur WebSphere Application Server Community Edition le service de données REST d'eXtreme Scale et l'exemple fourni :
   a. Ajoutez le fichier JAR d'exécution du client ObjectGrid au référentiel WebSphere Application Server Community Edition :
      1) Ouvrez et connectez-vous à la console d'administration de WebSphere Application Server Community Edition. (Les paramètres par défaut sont http://localhost:8080/console/ avec l'ID utilisateur system et le mot de passe manager.)
      2) Cliquez sur le lien **Référentiel** à gauche de la fenêtre de la console, dans le dossier Services.
      3) Dans la section **Ajouter une archive au référentiel**, entrez les éléments suivants dans les zones de texte :

Tableau 16. Ajout d'une archive au référentiel

Zone de texte	Valeur
Fichier	base_wxs/lib/ogclient.jar
Groupe	com.ibm.websphere.xs
Artefact	ogclient
Version	7.1
Type	JAR

      4) Cliquez sur le bouton Installer.

Conseil : Reportez-vous à la note technique suivante pour des détails sur les différentes manières dont les dépendances de classes et de bibliothèques peuvent être configurées : Specifying external dependencies to applications running on WebSphere Application Server Community Edition.

b. Déployez le module de service de données REST, `wxsrestservice.war`, vers le serveur WebSphere Application Server Community Edition.

1) Editez l'exemple de fichier XML de déploiement `base_servicerest/gettingstarted/wasce/geronimo-web.xml` pour inclure les dépendances de chemin d'accès dans les répertoires du chemin d'accès aux classes de l'exemple Mise en route :

- Modifiez le chemin "classesDirs" des deux GBeans du client Mise en route :

  Le chemin de "classesDirs" pour le bean géré GettingStarted_Client_SharedLib doit avoir la valeur `base_servicerest/gettingstarted/restclient/bin`.

  Le chemin de "classesDirs" pour le bean géré GettingStarted_Common_SharedLib doit avoir la valeur `base_servicerest/gettingstarted/restclient/bin`.

2) Ouvrez et connectez-vous à la console d'administration de WebSphere Application Server Community Edition.

3) Cliquez sur le lien **Déployer nouveau** situé dans la partie gauche de la fenêtre de la console.

4) Entrez les valeurs suivantes dans les zones de texte de la page **Installer de nouvelles applications** :

Tableau 17. Install New Applications

Zone de texte	Valeur
Archive	*base_servicerest*/lib/wxsrestservice.war
Plan	*base_servicerest*/gettingstarted/wasce/geronimo-web.xml

5) Cliquez sur le bouton **Installer**.

   La page de la console indique alors que l'application a été installée et démarrée.

6) Recherchez le message ci-après sur la console ou dans le journal de sortie système de WebSphere Application Server Community Edition pour vérifier que le service de données REST a bien démarré :

   `CWOBJ4000I: Le service de données REST de WebSphere eXtreme Scale a été démarré.`

8. Vérifiez que le service de données REST fonctionne :

   Ouvrez un navigateur Web et accédez à l'URL `http://<hôte>:<port>/<racine_contexte>/restservice/<nom_grille>`

   Le port par défaut de WebSphere Application Server Community Edition est 8080 et il est défini à l'aide de la propriété HTTPPort dans le fichier `/var/config/config-substitutions.properties`.

   Par exemple : http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/

Résultats

Le document de service AtomPub est affiché.

Déploiement du service de données REST sur Apache Tomcat

Nous allons expliquer comment configurer le service de données REST de WebSphere eXtreme Scale sur Apache Tomcat version 5.5 ou plus récente.

Pourquoi et quand exécuter cette tâche

- Un JRE ou JDK IBM ou Sun, version 5 ou ultérieure est installé et la variable d'environnement JAVA_HOME est spécifiée.
- Apache Tomcat Version 5.5 ou ultérieure est installé. Voir Apache Tomcat pour savoir comment installer Tomcat.
- La version d'évaluation d'eXtreme Scale version 7. avec le service de données REST est téléchargée et extraite ou le produit WebSphere eXtreme Scale version 7.1.0.0 avec le groupe de correctifs 2 est installé dans un répertoire autonome.

Procédure

1. Si vous utilisez un JDK ou un JRE Sun, installez l'ORB IBM dans Tomcat :
   a. Tomcat version 5.5 :

      Copiez tous les fichiers JAR depuis :

      le répertoire *base_wxs*/lib/endorsed

      vers :

      le répertoire *racine_tomcat*/common/endorsed
   b. Tomcat version 6.0 :

      Créez un répertoire "endorsed" :

      UNIX Linux `mkdir racine_tomcat/endorsed`

      Windows `md racine_tomcat/endorsed`

      Copiez tous les fichiers JAR de :

      `base_wxs/lib/endorsed`

      vers :

      `racine_tomcat/common/endorsed`
2. Configurez et démarrez une grille eXtreme Scale.
   a. Pour des explications détaillées sur la configuration d'une grille eXtreme Scale à utiliser avec le service de données REST, voir Chapitre 6, «Configuration de l'environnement de déploiement», à la page 97.
   b. Vérifiez qu'un client eXtreme Scale arrive à se connecter aux entités de la grille et à y accéder. Pour un exemple, voir Exemple et tutoriel sur les services de données REST.
3. Générez le répertoire ou le fichier JAR de configuration du service REST d'eXtreme Scale. Pour les détails, voir dans «Installation du service de données REST», à la page 303 les explications sur le packaging et le déploiement.
4. Déployez le module du service de données REST : wxsrestservice.war sur le serveur Tomcat.

   Copiez le fichier wxsrestservice.war depuis :

   `base_servicerest/lib`

   vers :

   `racine_tomcat/webapps`
5. Ajoutez le fichier JAR d'exécution du client ObjectGrid et le fichier JAR de l'application dans le chemin d'accès aux classes partagé, dans Tomcat :
   a. Modifiez le fichier `racine_tomcat/conf/catalina.properties`.
   b. Ajoutez les noms de chemin suivants à la fin de la propriété shared.loader, en les séparant par une virgule :
      - `base_wxs/lib/ogclient.jar`
      - `base_servicerest/gettingstarted/restclient/bin`

- `base_servicerest/gettingstarted/common/bin`

6. Si vous utilisez la sécurité Java 2, ajoutez les droits de sécurité au fichier de règles tomcat :
   - Si vous utilisez Tomcat version 5.5 :

     Fusionnez le contenu de l'exemple de fichier de `règles catalina` 5.5 qui se trouve dans

     `base_servicerest/gettingstarted/tomcat/catalina-5_5.policy` avec le fichier `racine_tomcat/conf/catalina.policy`.
   - Si vous utilisez Tomcat version 6.0 :

     Fusionnez le contenu de l'exemple de fichier de `règles catalina` 6.0 qui se trouve dans

     `base_servicerest/gettingstarted/tomcat/catalina-6_0.policy` avec le fichier `racine_tomcat/conf/catalina.policy`.
7. Démarrez le serveur Tomcat :
   - **Si vous utilisez Tomcat 5.5 sous UNIX or Windows, ou la distribution ZIP de Tomcat 6.0 :**
     a. `cd racine_tomcat/bin`
     b. Démarrez le serveur :
        - Sans la sécurité Java 2 activée :

          UNIX Linux `./catalina.sh run`

          Windows `catalina.bat run`
        - Avec la sécurité Java 2 activée :

          UNIX Linux `./catalina.sh run -security`

          Windows `catalina.bat run -security`

     c. Les journaux d'Apache Tomcat sont affichés sur la console. Lorsque le service de données REST a correctement démarré, le message suivant est affiché dans la console d'administration :

        `CWOBJ4000I: Le service de données REST de WebSphere eXtreme Scale a été démarré.`
   - **Si vous utilisez Tomcat 6.0 sous Windows à l'aide de la distribution du programme d'installation Windows :**
     a. cd /bin
     b. Démarrez l'outil de configuration d'Apache Tomcat 6 :

        `tomcat6w.exe`
     c. Pour activer la sécurité Java 2 : (facultatif) :

        Ajoutez les entrées suivantes aux options Java dans la page Java de la fenêtre des propriétés d'Apache Tomcat 6 :

        `-Djava.security.manager`

        `-Djava.security.policy=\conf\catalina.policy`
     d. Cliquez sur le bouton Démarrer de la fenêtre de propriétés d'Apache Tomcat 6 pour démarrer le serveur Tomcat.
     e. Consultez les journaux suivants pour vérifier que le serveur Tomcat a été correctement démarré :
        - `racine_tomcat/bin/catalina.log`

          Affiche le statut du moteur du serveur Tomcat
        - `racine_tomcat/bin/stdout.log`

          Affiche le journal de la sortie système.

f. Si le service de données REST est correctement démarré, le message suivant est affiché dans le journal de la sortie système :

   ```
   CWOBJ4000I: Le service de données REST de WebSphere eXtreme Scale a été démarré.
   ```

8. Vérifiez que le service de données REST fonctionne bien.

   Ouvrez un navigateur Web et accédez à l'adresse URL suivante :

   http://*host:port*/*racine_contexte*/restservice/*nom_grille*

   Le port par défaut de Tomcat est 8080 et il est configuré dans l'élément <Connector> du fichier racine_tomcat/conf/server.xml.

   Par exemple :

   ```
   http://localhost:8080/wxsrestservice/restservice/NorthwindGrid/
   ```

Résultats

Le document de service AtomPub est affiché.

Sécurisation du service de données REST

Vous pouvez sécuriser un bon nombre d'aspects du service de données REST. L'accès au service de données REST d'eXtreme Scale peut être sécurisé via l'authentification et l'autorisation. Il peut également être contrôlé par des règles de configuration de portée service, désignées sous le nom de règles d'accès. La sécurité des transports est le troisième élément concerné en matière de sécurisation.

Pourquoi et quand exécuter cette tâche

L'accès au service de données REST d'eXtreme Scale peut être sécurisé via l'authentification et l'autorisation. L'authentification et l'autorisation s'effectuent grâce à l'intégration à la sécurité d'eXtreme Scale.

L'accès peut également être contrôlé par des règles de configuration de portée service, désignées sous le nom de règles d'accès. Il existe deux types de règles d'accès : les droits d'opérations du service qui contrôlent les opérations CRUD autorisées par le service et les droits d'accès aux entités qui contrôlent les opérations CRUD autorisées pour un type donné d'entité.

La sécurité des transports est fournie par la configuration du conteneur hébergeant (pour les connexions du client Web au service REST) et par la configuration du client eXtreme Scale (pour les connexions du service REST à la grille eXtreme Scale).

Procédure

- Contrôlez l'authentification et l'autorisation.

  L'accès au service de données REST d'eXtreme Scale peut être sécurisé via l'authentification et l'autorisation. L'authentification et l'autorisation s'effectuent grâce à l'intégration à la sécurité d'eXtreme Scale.

  Le service de données REST d'eXtreme Scale utilise la sécurité d'eXtreme Scale (authentification et autorisation) pour contrôler quels sont les utilisateurs qui peuvent accéder au service et quelles opérations un utilisateur est autorisé à effectuer via le service. Le service de données REST d'eXtreme Scale utilise soit des données d'identification globales configurées (nom d'utilisateur et mot de passe), soit des données d'identification dérivées d'une demande d'authentification HTTP BASIC qui est envoyée avec chaque transaction à la grille eXtreme Scale où s'effectuent l'authentification et l'autorisation.

1. Configurez dans la grille l'authentification et l'autorisation des clients eXtreme Scale. Voir «Intégration de la sécurité à des fournisseurs externes», à la page 370 pour des explications détaillées sur la manière de configurer l'authentification et l'autorisation des clients eXtreme Scale.
2. Configurez la sécurité du client eXtreme Scale (utilisé par le service REST).

   Le service de données REST d'eXtreme Scale fait appel à la bibliothèque des clients eXtreme Scale lorsqu'il communique avec la grille eXtreme Scale. Il en résulte que le client eXtreme Scale doit être configuré pour la sécurité d'eXtreme Scale.

   L'authentification du client eXtreme Scale est activée via des propriétés dans le fichier des propriétés du client objectgrid. Au minimum, les attributs suivants doivent être activés lorsqu'on utilise la sécurité du client avec le service REST :

   ```
   securityEnabled=true
   credentialAuthentication=Supported [-ou-] Required
   credentialGeneratorProps=utilisateur:motdepasse [-ou-]
   {xor encoded utilisateur:motdepasse}
   ```

   Vous noterez que l'utilisateur et le mot de passe spécifiés dans la propriété credentialGeneratorProps doivent se mapper à un ID du registre d'authentification et qu'ils doivent disposer de droits de règles ObjectGrid suffisant pour pouvoir se connecter à des ObjectGrids et à en créer.

   Vous trouverez un exemple de fichier de règles de client objectgrid dans `base_wxsrest/security/security.ogclient.properties`. Voir également «Fichier de propriétés du client», à la page 205.
3. Configurez la sécurité du service de données REST d'eXtreme Scale.

   Le fichier des propriétés de configuration du service de données REST d'eXtreme Scale doit contenir les entrées suivantes pour que le service puisse s'intégrer à la sécurité d'eXtreme Scale :

   ```
   ogClientPropertyFile=nom_fichier
   ```

   ogClientPropertyFile est l'adresse du fichier de propriétés qui contient les propriétés du client ObjectGrid mentionnées au point précédent. Lorsque la sécurité est activée, le service REST utilise ce fichier pour initialiser le client eXtreme Scale afin de communiquer avec la grille.

   ```
   loginType=basic [-ou-] none
   ```

   La propriété loginType configure le service REST pour le type d'ouverture de session. Si "none" est spécifié comme valeur, l'ID utilisateur et le mot de passe "globaux" définis par credentialGeneratorProps seront envoyés à la grille à l'occasion de chaque transaction. Si c'est une valeur "basic" qui est spécifiée, le service REST présentera au client une demande d'authentification HTTP BASIC en réclamant des données d'identification qu'il enverra dans chaque transaction lorsqu'il communiquera avec la grille.

   Pour plus d'informations sur les propriétés ogClientPropertyFile et loginType, voir «Fichier de propriétés du service de données REST», à la page 290.

- Appliquez des règles d'accès.

  L'accès peut également être contrôlé par des règles de configuration de portée service, désignées sous le nom de règles d'accès. Il existe deux types de règles d'accès : les droits d'opérations du service qui contrôlent les opérations CRUD autorisées par le service et les droits d'accès aux entités qui contrôlent les opérations CRUD autorisées pour un type donné d'entité.

  Le service de données REST d'eXtreme Scale autorise, si on le souhaite, des règles d'accès configurables à accéder de manière restreinte au service et aux entités contenues dans ce dernier. Ces règles d'accès sont spécifiées dans le

fichier des propriétés des droits d'accès du service REST. Le nom de ce fichier est spécifié dans le fichier des propriétés du service par la propriété wxsRestAccessRightsFile (voir la section 5.1). Ce fichier est un fichier de propriétés Java classique avec des paires clé/valeur. Il existe deux types de règles d'accès : les droits d'opérations du service qui contrôlent les opérations CRUD autorisées par le service et les droits d'accès aux entités qui contrôlent les opérations CRUD autorisées pour un type donné d'entité.

1. Configurez les droits d'opérations du service.

   Les droits d'opérations du service spécifient les droits d'accès qui s'appliquent à tous les ObjectGrids exposés via le service REST ou à toutes les entités de l'ObjectGrid individuel qui est spécifié.

   Utilisez la syntaxe suivante.

   ```
   serviceOperationRights=droit_opérations_service
   serviceOperationRights.nom_grille -ou- *=droit_opérations_service
   ```

   où

   – serviceOperationRights peut être l'un des suivants : [NONE, READSINGLE, READMULTIPLE, ALLREAD, ALL]

   – serviceOperationRights.*nom_grille -ou-* * implique que le droit d'accès s'applique à tous les ObjectGrids, autrement le nom d'un ObjectGrid spécifique peut être fourni.

   Par exemple :

   ```
   serviceOperationsRights=ALL
   serviceOperationsRights.*=NONE
   serviceOperationsRights.EMPLOYEEGRID=READSINGLE
   ```

   Le premier exemple spécifie que toutes les opérations du service sont autorisées pour tous les ObjectGrids exposés par ce service REST. Le deuxième exemple est semblable au premier car il s'applique également à tous les ObjectGrids exposés par le service REST, mais il spécifie des droits d'accès NONE, ce qui signifie qu'aucune opération du service n'est autorisée sur les ObjectGrids. Le dernier exemple spécifie comment contrôler les opérations du service pour une grille spécifique ; ici seules les opérations de lecture qui donnent un seul enregistrement sont autorisées pour toutes les entités de la grille EMPLOYEEGRID.

   Par défaut, le service REST présuppose que serviceOperationsRights=ALL, autrement dit, que toutes les opérations sont autorisées pour tous les ObjectGrids exposés par ce service. C'est l'inverse de l'implémentation de Microsoft® qui a choisi NONE comme valeur par défaut, interdisant du coup toute opération au service REST.

   Remarque : Les droits d'opérations du service sont évalués dans l'ordre dans lequel ils sont spécifiés dans ce fichier, ce qui fait que le dernier droit à être spécifié prendra le pas sur les droits qui viennent avant lui.

2. Configurez les droits d'accès aux entités.

   Les droits d'ensembles d'entités spécifient les droits d'accès qui s'appliquent aux entités de l'ObjectGrid spécifique qui est exposé via le service REST. Ces droits permettent d'imposer un contrôle bien plus étroit et bien plus granulaire de l'accès à des entités d'un ObjectGrid individuel que ne le permettent les droits d'opérations du service.

   Utilisez la syntaxe suivante.

   ```
   entitySetRights.nom_grille.nom_entité=droit_ensemble_entités
   ```

   où

– *droit_ensemble_entités* peut être l'un des droits suivants

Tableau 18. Droits d'accès à des entités. Valeurs prises en charge.

Droit d'accès	Description
NONE	Refuse tout droit d'accès aux données
READSINGLE	Autorise la lecture d'un seul élément de données
READMULTIPLE	Autorise la lecture d'ensembles de données
ALLREAD	Autorise toutes les opérations de lecture (élément simple ou ensembles de données)
WRITEAPPEND	Autorise la création de nouveaux éléments de données dans les ensembles de données
WRITEREPLACE	Autorise le remplacement de données
WRITEDELETE	Autorise la suppression d'éléments de données dans les ensembles de données
WRITEMERGE	Autorise la fusion de données
ALLWRITE	Autorise toutes les opérations d'écriture (création, remplacement, fusion ou suppression) de données
ALL	Autorise la création, la lecture, la modification et la suppression de données

– *nom_entité* est le nom d'un ObjectGrid spécifique au sein du service REST
– *nom_grille* est le nom d'une entité spécifique au sein de l'ObjectGrid spécifié

Remarque : Si les droits d'opérations du service et les droits d'ensembles d'entités sont spécifiés en même temps pour un ObjectGrid et ses entités, le droit appliqué sera le plus restrictif des deux, comme le montrent les exemples qui suivent. Rappelez-vous également que les droits d'ensembles d'entités sont évalués dans l'ordre où ils sont spécifiés dans le fichier. Le dernier droit à être spécifié prendra le pas sur ceux qui viennent avant lui.

Exemple 1 : Si serviceOperationsRights.NorthwindGrid=READSINGLE et entitySetRights.NorthwindGrid.Customer=ALL sont spécifiés. READSINGLE sera appliqué pour l'entité Customer.

Exemple 2 : Si serviceOperationsRights.NorthwindGrid=ALLREAD est spécifié et qu'entitySetRights.NorthwindGrid.Customer=ALLWRITE l'est aussi, seules des opérations de lecture seront autorisées pour toutes les entités de NorthwindGrid. Mais, en ce qui concerne Customer, ses droits d'ensembles d'entités empêcheront toute lecture (puisque c'est ALLWRITE qui est spécifié) et de ce fait l'entité Customer aura NONE comme droit d'accès.

- Sécurisez les transports.

  La sécurité des transports est fournie par la configuration du conteneur hébergeant (pour les connexions du client Web au service REST) et par la configuration du client eXtreme Scale (pour les connexions du service REST à la grille eXtreme Scale).

  1. Sécurisez la connexion entre le client et le service REST. La sécurité des transports pour cette connexion est fournie par l'environnement du conteneur hébergeant et non dans eXtreme Scale.
  2. Sécurisez la connexion entre le service REST et la grille eXtreme Scale. La sécurité des transports pour cette connexion est configurée dans eXtreme Scale. Voir «Protocole TLS et couche de connexion sécurisée», à la page 365.

Chapitre 7. Exploitation de l'environnement de déploiement

L'exploitation de l'environnement du produit consiste à démarrer et à arrêter les serveurs en mode autonome ou dans WebSphere Application Server. Vous pouvez également utiliser WebSphere eXtreme Scale comme gestionnaire de sessions dans un environnement WebSphere Application Server.

Terminologie des tâches administratives

Avant de vous lancer dans l'administration de WebSphere eXtreme Scale, il convient de se familiariser avec les faits suivants.

Pourquoi et quand exécuter cette tâche

Types de serveur

WebSphere eXtreme Scale possède deux types de serveur : les *serveurs de catalogues* et les *serveurs conteneurs*. Les serveurs de catalogues contrôlent le positionnement des fragments et détectent et surveillent les serveurs conteneurs. Ensembles, plusieurs serveurs de catalogues constituent le *service de catalogue*. Les serveurs conteneurs sont les machines virtuelles Java qui stockent les données d'application de la grille.

Mode autonome

Le mode autonome correspond à une configuration de WebSphere eXtreme Scale exécutée seule, sans aucun autre serveur d'applications.

Exécution dans WebSphere Application Server

Lorsque vous exécutez WebSphere eXtreme Scale par dessus WebSphere Application Server, les serveurs de catalogues sont démarrés automatiquement sur les serveurs WebSphere Application Server. Pour démarrer les serveurs conteneurs, vous devez packager et déployer votre application avec les fichiers XML d'ObjectGrid.

Conteneurs, partitions et fragments

Le conteneur est un service qui stocke les données d'application pour la grille. Ces données sont généralement fractionnées en différentes parties appelées partitions et hébergées par plusieurs conteneurs. A son tour, chaque conteneur héberge un sous-ensemble de données. Une machine virtuelle Java peut héberger un ou plusieurs conteneurs et chaque conteneur peut héberger plusieurs fragments.

A faire : Planifiez la taille des segments de mémoire des conteneurs qui hébergent l'ensemble de vos données. Configurez en conséquence les paramètres du segment de mémoire.

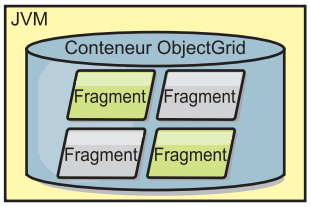


Figure 19. Conteneur

Les partitions hébergent un sous-ensemble des données dans la grille. WebSphere eXtreme Scale place automatiquement plusieurs partitions dans un même conteneur et les répartit différemment au fur et à mesure que des conteneurs deviennent disponibles.

Important : Choisissez soigneusement le nombre de partitions avant le déploiement final, car ce nombre ne peut pas être modifié dynamiquement. Un mécanisme de hachage permet de localiser les partitions dans le réseau et eXtreme Scale ne peut pas hacher à nouveau l'ensemble des données après leur déploiement. En règle générale, vous pouvez surestimer le nombre de partitions

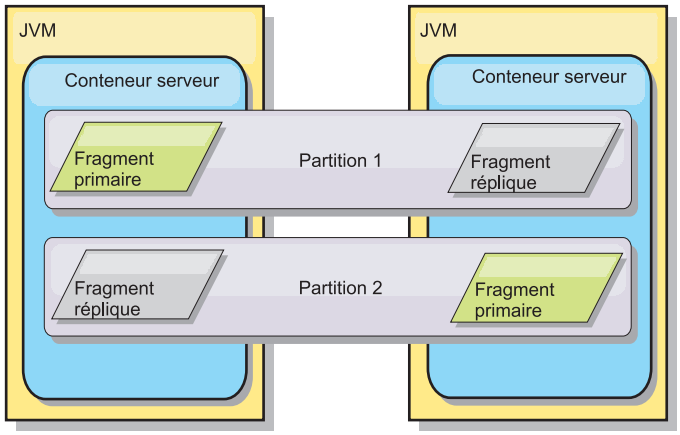


Figure 20. Partition

Les fragments sont des instances de partitions. Ils peuvent avoir un rôle primaire ou un rôle de réplique. Le fragment primaire et ses fragments répliques constituent la manifestation physique de la partition. Chaque partition contient plusieurs fragments, dont chacun héberge toutes les données contenues dans celle-ci. L'un des fragments est le fragment primaire, les autres sont les fragments répliques, c'est-à-dire des copies redondantes des données du fragment primaire. Le fragment primaire est la seule instance de partition permettant à des transactions d'écrire dans le cache. Un fragment réplique est une instance "miroir" de la partition. Il reçoit des mises à jour du fragment primaire de manière synchrone ou asynchrone. Le fragment réplique autorise uniquement les transactions à lire à partir du cache. Les fragments répliques ne sont jamais hébergés dans le même conteneur ni sur la même machine que le fragment primaire.

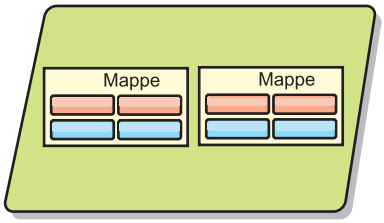


Figure 21. Fragment

Pour améliorer la disponibilité des données ou garantir la persistance, répliquez les données. La réplication augmente toutefois le coût des transactions et améliore les performances au détriment de la disponibilité. Avec eXtreme Scale, vous pouvez contrôler les coûts car les réplications synchrones et asynchrones sont prises en charge, ainsi que les modèles de réplication hybrides utilisant les deux modes de réplication. Un fragment réplique synchrone reçoit des mises à jour lors de la transaction du fragment primaire visant à garantir la cohérence des données. Un fragment réplique synchrone peut doubler le temps de réponse car la transaction doit valider le fragment primaire et le fragment réplique synchrone avant que la transaction se termine. Un fragment réplique asynchrone reçoit des mises à jour après que la transaction a validé la limitation de l'impact sur les performances, mais introduit la possibilité de perte de données car les fragments réplique asynchrones peuvent impliquer le traitement de plusieurs transactions après le fragment primaire.

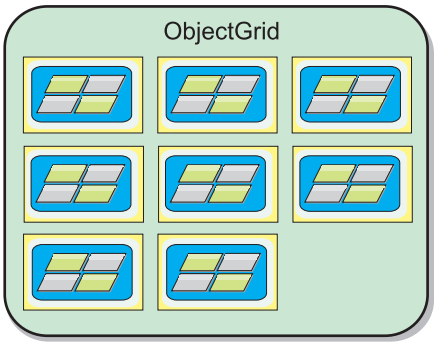


Figure 22. ObjectGrid

Services de catalogue (serveurs de catalogue)

Le service de catalogue héberge une logique qui doit être inactive lorsque l'état est stabilisé et qui a peu d'influence sur l'évolutivité. Le service de catalogue est généré pour gérer plusieurs centaines de conteneurs devenant disponibles simultanément. Il exécute des services en vue de leur gestion.

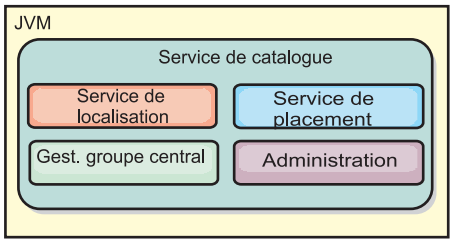


Figure 23. Service de catalogue

Les responsabilités du catalogue de service sont les suivantes :

Service de localisation
Le service de localisation fournit une localité aux clients qui recherchent des conteneurs hébergeant des applications et aux conteneurs cherchant à enregistrer des applications hébergées à l'aide du service de positionnement. Il s'exécute dans tous les membres de la grille et permet de supprimer cette fonction.

Service de positionnement
Le service de positionnement constitue le système nerveux central de la grille. Il est responsable de l'allocation des fragments à leur conteneur hôte. Il s'exécute en tant que service Un sur N choisi dans le cluster. Comme c'est la règle Un sur N qui est utilisée, il y a toujours une et une seule instance de ce service en cours d'exécution. Si cette instance doit s'arrêter, un autre processus prend la relève. A des fins de redondance, tous les états du service de catalogue sont répliqués sur tous les serveurs hébergeant le service de catalogue.

Gestionnaire du groupe central
Le gestionnaire du groupe central gère le regroupement homologue en vue de la surveillance de la santé, organise les conteneurs en petits groupes de serveurs et fédère automatiquement les groupes de serveurs. Lorsqu'un conteneur contacte le service de catalogue pour la première fois, le conteneur attend d'être affecté à un nouveau groupe ou à un groupe existant de plusieurs machines virtuelles Java. Chaque groupe de machines virtuelles Java surveille la disponibilité de chacun de ses membres à l'aide du signal de présence. L'un des membres du groupe relaie les informations sur la disponibilité au service de catalogue afin de lui permettre de réagir aux défaillances en procédant à des réallocations et à des réacheminements.

Administration
L'administration de l'environnement WebSphere eXtreme Scale se compose de quatre phases : la planification, le déploiement, la gestion et la surveillance. Pour plus d'informations sur chacune de ces étapes, consultez le manuel *Guide d'administration*.

Pour la disponibilité, configurez un domaine de services de catalogue. Un domaine de services de catalogue consiste en plusieurs machines virtuelles Java, dont une machine maîtresse et plusieurs machines virtuelles Java de sauvegarde.

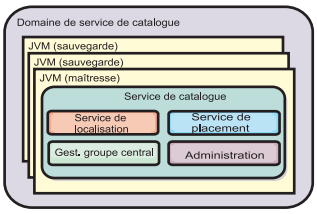


Figure 24. Domaine de services de catalogue

Définition de la disponibilité d'un ObjectGrid

L'état de disponibilité d'une instance ObjectGrid détermine les requêtes pouvant être traitées à tout moment.

Il existe quatre états de disponibilité :

- ONLINE
- QUIESCE
- OFFLINE
- PRELOAD

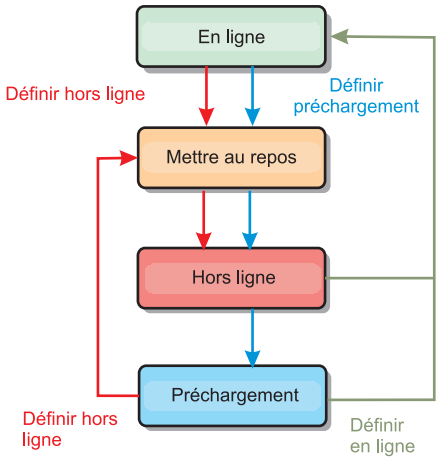


Figure 25. Etats de disponibilité d'un ObjectGrid

Définition de l'état de disponibilité

L'état de disponibilité par défaut d'un ObjectGrid est ONLINE (en ligne). Un ObjectGrid en ligne est capable de traiter n'importe quelle requête d'un client eXtreme Scale typique. Toutefois, les requêtes d'un client de préchargement sont rejetées lorsque l'ObjectGrid est en ligne.

L'état QUIESCE (au repos) est transitionnel. Un ObjectGrid au repos passe rapidement à l'état OFFLINE (hors ligne). Un ObjectGrid au repos est autorisé à traiter des transactions en attente. Par contre, toute nouvelle transaction est rejetée. Un ObjectGrid peut rester au repos jusqu'à 30 secondes, après quoi l'état de disponibilité passe en OFFLINE.

Un ObjectGrid hors ligne rejette toutes les transactions.

L'état PRELOAD (préchargement) peut servir à charger des données dans un ObjectGrid à partir d'un client de préchargement. Lorsque l'ObjectGrid est à l'état de préchargement, seul un client de préchargement peut valider des transactions par rapport à cet ObjectGrid. Toutes les autres transactions sont rejetées

Utilisez l'interface de StateManager pour définir l'état de disponibilité d'un ObjectGrid. Pour définir l'état de disponibilité d'un ObjectGrid exécuté sur les serveurs, transmettez un client ObjectGrid correspondant à l'interface de StateManager. Le code suivant démontre comment changer l'état de disponibilité d'un ObjectGrid.

```
Client ClientClusterContext = ogManager.connect("localhost:2809", null, null);
ObjectGrid myObjectGrid = ogManager.getObjectGrid(client, "myObjectGrid");
StateManager stateManager = StateManagerFactory.getStateManager();
stateManager.setObjectGridState(AvailabilityState.OFFLINE, myObjectGrid);
```

Chaque fragment de l'ObjectGrid passe à l'état à appliquer lorsque la méthode setObjectGridState est appelée sur l'interface de StateManager. Lorsque la méthode est renvoyée, tous les fragments de l'ObjectGrid doivent être définis sur l'état adéquat.

Utilisez un plug-in ObjectGridEventListener pour changer l'état de disponibilité d'un ObjectGrid côté serveur. Changez l'état de disponibilité d'un ObjectGrid côté serveur seulement lorsque ce dernier présente une partition unique. Si l'ObjectGrid présente plusieurs partitions, la méthode shardActivated est appelée sur chaque partition principale, ce qui entraîne des appels superflus pour le changement d'état de l'ObjectGrid

```
public class OGListener implements ObjectGridEventListener,
   ObjectGridEventGroup.ShardEvents {
    public void shardActivated(ObjectGrid grid) {
            StateManager stateManager = StateManagerFactory.getStateManager();
            stateManager.setObjectGridState(AvailabilityState.PRELOAD, grid);
    }
}
```

L'état QUIESCE étant transitionnel, vous ne pouvez pas utiliser l'interface de StateManager pour définir l'état d'un ObjectGrid sur QUIESCE. L'ObjectGrid passe par cet état avant d'être défini sur l'état OFFLINE.

Récupération de l'état de disponibilité

Utilisez la méthode getObjectGridState de l'interface de StateManager pour récupérer l'état de disponibilité d'un ObjectGrid.

```
StateManager stateManager = StateManagerFactory.getStateManager();
AvailabilityState state = stateManager.getObjectGridState(inventoryGrid);
```

La méthode getObjectGridState choisit une partition principale de l'ObjectGrid au hasard et renvoie son état de disponibilité. Tous les fragments ObjectGrid doivent présenter le même état ou être en transition vers le même état. C'est pour cela que cette méthode propose un résultat acceptable pour l'état de disponibilité actuel de l'ObjectGrid.

Etats de disponibilité appropriés pour diverses requêtes

Une requête est rejetée lorsque l'état d'un ObjectGrid n'est pas approprié pour traiter cette requête. Dans ce cas, une exception AvailabilityException est générée.

Attribut initialState

Vous pouvez utiliser l'attribut initialState avec un ObjectGrid pour indiquer son état au démarrage. Normalement, lorsqu'un ObjectGrid termine son initialisation, il devient disponible pour le routage. L'état peut ensuite être changé de manière à empêcher l'acheminement du trafic vers l'ObjectGrid. Si l'ObjectGrid doit être initialisé sans être rendu disponible immédiatement, vous pouvez utiliser l'attribut initialState.

L'attribut initialState est défini dans le fichier XML de configuration de l'ObjectGrid. L'état par défaut est ONLINE. Les valeurs admises sont les suivantes :

- ONLINE (par défaut)
- PRELOAD
- OFFLINE

Pour plus d'informations, voir la documentation de l'API AvailabilityState.

Si l'attribut initialState est défini pour un ObjectGrid, l'état doit être explicitement redéfini comme en ligne. Sinon, l'ObjectGrid reste indisponible et émet des exceptions de disponibilité.

Utilisation de l'attribut initialState pour le préchargement

Si l'ObjectGrid est préchargé avec des données, un laps de temps est susceptible de s'écouler entre le moment où il est disponible et le moment où il passe à l'état de préchargement permettant de bloquer le trafic client. Pour éviter ce laps de temps, l'état initial d'un ObjectGrid peut être défini comme PRELOAD. L'ObjectGrid effectue toujours l'initialisation requise, mais bloque le trafic jusqu'au changement d'état et permet au préchargement d'avoir lieu.

Les états PRELOAD et OFFLINE bloquent le trafic, mais seul l'état PRELOAD permet de lancer un préchargement.

Basculement et équilibrage

Si un fragment réplique est promu au rang de fragment primaire, il n'utilise pas l'attribut initialState. Si le fragment primaire est déplacé pour cause de rééquilibrage, l'attribut initialState ne sera pas utilisé, car les données sont copiées vers le nouvel emplacement du fragment primaire avant la fin du déplacement. Si la réplication n'est pas configurée, le fragment primaire prend la valeur initialState en cas de basculement et un nouveau fragment primaire doit être positionné.

Utilisation de l'API des serveurs imbriqués

Avec WebSphere eXtreme Scale, vous pouvez utiliser une interface de programmation d'application pour gérer le cycle de vie des conteneurs et serveurs imbriqués. Vous pouvez configurer le serveur à l'aide d'un programme avec les options que vous pouvez également configurer avec la ligne de commande ou les propriétés de serveur incluses dans un fichier. Vous pouvez configurer le serveur imbriqué pour en faire un serveur conteneur et/ou un service de catalogue.

Avant de commencer

Vous devez disposer d'une méthode permettant d'exécuter du code depuis une machine virtuelle Java existante. les classes eXtreme Scale doivent être disponibles dans l'arborescence du chargeur de classe.

Pourquoi et quand exécuter cette tâche

Vous pouvez effectuer de nombreuses tâches d'administration à l'aide de l'API d'administration. L'API est couramment utilisée comme serveur interne pour stocker l'état d'une application Web. Le serveur Web peut démarrer un serveur WebSphere eXtreme Scale imbriqué et signaler le serveur conteneur au service de catalogue. Le serveur est ensuite ajouté comme membre d'une grille répartie plus importante. Cette utilisation peut offrir des possibilité d'évolution et une haute disponibilité à un fichier de données qui reste sinon volatile.

Vous pouvez contrôler à l'aide d'un programme le cycle de vie complet d'un serveur eXtreme Scale imbriqué. Les exemples sont aussi génériques que possible et n'illustrent que des exemples de code spécifiques aux étapes présentées.

Procédure

1. Procurez-vous l'objet ServerProperties de la classe ServerFactory et configurez les options nécessaires.

   Chaque serveur eXtreme Scale possède un ensemble de propriétés configurables. Lorsqu'un serveur est démarré à partir de la ligne de commande, ces propriétés reçoivent les valeurs par défaut, mais vous pouvez remplacer plusieurs propriétés en fournissant un fichier ou une source externe. Dans la portée imbriquée, vous pouvez directement définir les propriétés avec un objet ServerProperties. Vous devez définir ces propriétés avant d'obtenir une instance de serveur de la classe ServerFactory. L'exemple de fragment de code ci-après obtient un objet ServerProperties, définit le champ CatalogServiceBootStrap et initialise plusieurs paramètres de serveur facultatifs. Pour une liste des paramètres configurables, reportez-vous à la documentation de l'API.

   ```
   ServerProperties props = ServerFactory.getServerProperties();
   props.setCatalogServiceBootstrap("host:port"); // requis pour se connecter à un service
                                                  de catalogue spécifique
   props.setServerName("ServerOne"); // nommez le serveur
   props.setTraceSpecification("com.ibm.ws.objectgrid=all=enabled"); // Définit la
    spécification de trace
   ```

2. Si vous souhaitez que le serveur soit un service de catalogue, procurez-vous l'objet CatalogServerProperties.

   Chaque serveur imbriqué peut être un service de catalogue, un serveur conteneur ou un serveur conteneur et un service de catalogue. L'exemple ci-après obtient l'objet CatalogServerProperties, active l'option de service de catalogue et configure divers paramètres de service de catalogue.

   ```
   CatalogServerProperties catalogProps = ServerFactory.getCatalogProperties();
   catalogProps.setCatalogServer(true); // false par défaut ; doit être défini comme
                                        service de catalogue
   catalogProps.setQuorum(true); // active/désactive le quorum
   ```

3. Procurez-vous une instance Server à partir de la classe ServerFactory. L'instance Server est un singleton de portée processus chargé de gérer l'appartenance dans la grille. Une fois que cette instance a été instanciée, ce processus est connecté et devient hautement disponible pour les autres serveurs de la grille. L'exemple suivant montre comment créer l'instance Server :

   ```
   Server server = ServerFactory.getInstance();
   ```

   Si nous considérons l'exemple précédent, la classe ServerFactory fournit une méthode statique qui renvoie une instance Server. La classe ServerFactory est prévue pour être la seule interface permettant d'obtenir une instance Server. Par conséquent la classe garantit que l'instance est un singleton ou une instance pour chaque machine virtuelle Java ou chargeur de classe isolé. La méthode getInstance initialise l'instance Server. Vous devez configurer toutes les

propriétés du serveur avant d'initialiser l'instance. La classe Server est chargée de créer des instances Container. Vous pouvez utiliser à la fois la classe ServerFactory et la classe Server pour gérer le cycle de vie de l'instance Server imbriquée.

4. Démarrez une instance Container à l'aide de l'instance Server.

   Pour que des fragments puissent être positionnées sur un serveur imbriqué, vous devez créer un conteneur sur le serveur. L'interface Server contient une méthode createContainer et accepte un argument DeploymentPolicy. L'exemple ci-après utilise l'instance de serveur que vous avez obtenue pour créer un conteneur à l'aide du fichier de règles de déploiement. Notez que les conteneurs requièrent un chargeur de classe pour lequel les fichiers binaires de l'application sont disponibles, à des fins de sérialisation. Vous pouvez rendre ces fichiers binaires disponibles en appelant la méthode createContainer avec comme chargeur de classe du contexte de l'unité d'exécution, celui que vous souhaitez utiliser.

   ```
   DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy(new
      URL("file://urltodeployment.xml"),
    new URL("file://urltoobjectgrid.xml"));
   Container container = server.createContainer(policy);
   ```

5. Supprimez et nettoyez un conteneur.

   Vous pouvez supprimer et nettoyer un serveur conteneur en exécutant la méthode teardown sur l'instance Container obtenue. L'exécution de la méthode teardown sur un conteneur nettoie ce dernier de manière appropriée et supprime le conteneur du serveur imbriqué.

   La procédure de nettoyage du conteneur inclut le déplacement et le démontage de tous les fragments positionnés dans ce conteneur. Chaque serveur peut contenir plusieurs conteneurs et fragments. Le nettoyage d'un conteneur n'affecte pas le cycle de vie de l'instance Server parent. L'exemple ci-après montre comment exécuter la méthode teardown sur un serveur. La méthode teardown est rendue accessible via l'interface ContainerMBean. En utilisant l'interface ContainerMBean, si vous n'avez plus accès à ce conteneur à l'aide d'un programme, vous pouvez toujours le supprimer et le nettoyer avec son bean géré. Une méthode terminate existe également dans l'interface Container ; ne l'utilisez pas, sauf si cela est indispensable. Cette méthode est plus puissante et ne coordonne pas un déplacement et un nettoyage des fragments appropriés.

   ```
   container.teardown();
   ```

6. Arrêtez le serveur imbriqué.

   Lorsque vous arrêtez un serveur imbriqué, vous arrêtez également les conteneurs et les fragments en cours d'exécution sur ce serveur. Lorsque vous arrêtez un serveur imbriqué, vous devez nettoyer toutes les connexions ouvertes et déplacer ou démonter tous les fragments. L'exemple ci-après illustre comment arrêter un serveur et utiliser la méthode waitFor sur l'instance Server pour s'assurer que cette dernière s'arrête complètement. Comme pour l'exemple de conteneur, la méthode stopServer est rendue accessible via l'interface ServerMBean. A l'aide de cette interface, vous pouvez arrêter un serveur avec le bean géré (MBean) correspondant.

   ```
   ServerFactory.stopServer();  // Utilise la fabrique pour arrêter le singleton du serveur
    // ou
   server.stopServer(); // Utilise directement l'instance Server
   server.waitFor(); // Est renvoyé une fois que le serveur a correctement terminé
                       ses procédures d'arrêt
   ```

   Exemple de code complet :

   ```
    import java.net.MalformedURLException;
   import java.net.URL;

   import com.ibm.websphere.objectgrid.ObjectGridException;
   import com.ibm.websphere.objectgrid.deployment.DeploymentPolicy;
   ```

```
import com.ibm.websphere.objectgrid.deployment.DeploymentPolicyFactory;
import com.ibm.websphere.objectgrid.server.Container;
import com.ibm.websphere.objectgrid.server.Server;
import com.ibm.websphere.objectgrid.server.ServerFactory;
import com.ibm.websphere.objectgrid.server.ServerProperties;

public class ServerFactoryTest {

    public static void main(String[] args) {

        try {

            ServerProperties props = ServerFactory.getServerProperties();
            props.setCatalogServiceBootstrap("catalogservice-hostname:catalogservice-port");
            props.setServerName("ServerOne"); // name server
            props.setTraceSpecification("com.ibm.ws.objectgrid=all=enabled"); // TraceSpec

            /*
             * Dans la plupart des cas, le serveur ne sert que de serveur conteneur
             * et se connecte à un service de catalogue externe. Cette utilisation
             * favorise davantage la haute disponibilité. L'extrait de code commenté
             * ci-après permet à ce serveur de devenir un service de catalogue.
             *
             *
             * CatalogServerProperties catalogProps =
             * ServerFactory.getCatalogProperties();
             * catalogProps.setCatalogServer(true); // activez le service de catalogue
             * catalogProps.setQuorum(true); // activez le quorum
             */

            Server server = ServerFactory.getInstance();

            DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy
     (new URL("url to deployment xml"),  new URL("url to objectgrid xml file"));
            Container container = server.createContainer(policy);

            /*
             * Le fragment est maintenant positionné sur ce conteneur si les exigences
             * de déploiement sont satisfaites.
             * Cela englobe la création du serveur et du conteneur imbriqués.
             *
             * Les lignes ci-après illustrent simplement l'appel des méthodes de nettoyage
             */

            container.teardown();
            server.stopServer();
            int success = server.waitFor();

        } catch (ObjectGridException e) {
            // Container failed to initialize
        } catch (MalformedURLException e2) {
            // invalid url to xml file(s)
        }

    }

}
```

API de serveurs intégrés

WebSphere eXtreme Scale comprend des interfaces de programmes d'application (API) et des interfaces de programmation de système permettant d'intégrer des serveurs et des clients eXtreme Scale dans vos applications Java existantes. Nous allons décrire ces API de serveurs intégrés.

Instanciation du serveur eXtreme Scale

Plusieurs propriétés permettent de configurer l'instance du serveur eXtreme Scale, qu'il est possible d'extraire de la méthode ServerFactory.getServerProperties. L'objet ServerProperties étant un singleton, chaque appel à la méthode getServerProperties extrait la même instance.

Le code suivant permet de créer un serveur :

```
Server server = ServerFactory.getInstance();
```

Toutes les propriétés définies avant la première invocation de getInstance sont utilisées pour initialiser le serveur.

Définir les propriétés du serveur

Vous pouvez définir les propriétés du serveur jusqu'au premier appel à la méthode ServerFactory.getInstance. Ce premier appel instancie le serveur eXtreme Scale et lit toutes les propriétés configurées. Les propriétés définies après la création n'ont aucun effet. L'exemple qui suit montre comment définir les propriétés avant d'instancier une instance Server.

```
// L'on obtient les propriétés du serveur associées à ce processus.
ServerProperties serverProperties = ServerFactory.getServerProperties();

// L'on définit le nom du serveur pour ce processus.
serverProperties.setServerName("EmbeddedServerA");

// L'on définit le nom de la zone dans laquelle est contenu ce processus.
serverProperties.setZoneName("EmbeddedZone1");

// L'on définit les informations de point de contact requises pour l'amorçage
// du service de catalogue.

serverProperties.setCatalogServiceBootstrap("localhost:2809");

// L'on définit le nom de l'hôte d'écoute de l'ORB à utiliser pour la liaison.
serverProperties.setListenerHost("host.local.domain");

// L'on définit le port d'écoute de l'ORB à utiliser pour la liaison.
serverProperties.setListenerPort(9010);

// L'on désactive tous les beans gérés pour ce processus.
serverProperties.setMBeansEnabled(false);

Server server = ServerFactory.getInstance();
```

Incorporer le service de catalogue

Tout paramètre JVM marqué par la méthode CatalogServerProperties.setCatalogServer peut héberger le service de catalogue d'eXtreme Scale. Cette méthode demande à l'environnement d'exécution du serveur eXtreme Scale d'instancier le service de catalogue lors du démarrage du serveur. Le code qui suit montre comment instancier le serveur de catalogue eXtreme Scale :

```
CatalogServerProperties catalogServerProperties =
 ServerFactory.getCatalogProperties();
catalogServerProperties.setCatalogServer(true);

Server server = ServerFactory.getInstance();
```

Incorporer le conteneur eXtreme Scale

La méthode Server.createContainer permet à n'importe quelle machine virtuelle Java d'héberger plusieurs conteneurs eXtreme Scale. Le code qui suit montre comment instancier un conteneur eXtreme Scale :

```
Server server = ServerFactory.getInstance();
DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy(
    new File("META-INF/embeddedDeploymentPolicy.xml").toURI().toURL(),
    new File("META-INF/embeddedObjectGrid.xml").toURI().toURL());
Container container = server.createContainer(policy);
```

Processus serveur autonome

Vous pouvez démarrer ensemble tous les services, ce qui est utilise aussi bien en phase de développement qu'en production. En démarrant les services ensemble, le même processus se charge de toutes les tâches suivantes : démarrage du service de catalogue, démarrage d'un ensemble de conteneurs, exécution de la logique de connexion client. Démarrer les services de cette manière résout les problèmes de programmation antérieurs au déploiement dans un environnement réparti. Le code qui suit montre comment instancier un serveur eXtreme Scale autonome :

```
CatalogServerProperties catalogServerProperties =
 ServerFactory.getCatalogProperties();
catalogServerProperties.setCatalogServer(true);

Server server = ServerFactory.getInstance();
DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy(
    new File("META-INF/embeddedDeploymentPolicy.xml").toURI().toURL(),
    new File("META-INF/embeddedObjectGrid.xml").toURI().toURL());
Container container = server.createContainer(policy);
```

Intégrer eXtreme Scale dans WebSphere Application Server

La configuration d'eXtreme Scale s'effectue automatiquement lorsqu'on installe WebSphere Extended Deployment DataGrid dans un environnement WebSphere Application Server. Vous n'êtes pas obligé de définir des propriétés avant d'accéder au serveur pour créer un conteneur. Le code qui suit montre comment instancier un serveur eXtreme Scale dans WebSphere Application Server :

```
Server server = ServerFactory.getInstance();
DeploymentPolicy policy = DeploymentPolicyFactory.createDeploymentPolicy(
    new File("META-INF/embeddedDeploymentPolicy.xml").toURI().toURL(),
    new File("META-INF/embeddedObjectGrid.xml").toURI().toURL);
Container container = server.createContainer(policy);
```

«Utilisation de l'API des serveurs imbriqués», à la page 325 contient un exemple expliquant pas à pas comment démarrer par programmation un service de catalogue et un conteneur intégrés.

Démarrage de serveurs WebSphere eXtreme Scale autonomes

Quand vous exécutez une configuration WebSphere eXtreme Scale autonome, l'environnement se compose de serveurs de catalogues, de serveurs conteneurs et de processus de client eXtreme Scale. Des serveurs eXtreme Scale peuvent également avoir été imbriqués dans des applications Java existantes à l'aide de l'API de serveur intégré. Vous devez manuellement configurer et démarrer ces processus.

Avant de commencer

Vous pouvez démarrer des serveurs WebSphere eXtreme Scale dans un environnement dans lequel WebSphere Application Server n'est pas installé. Si vous utilisez WebSphere Application Server, voir «Administration de WebSphere eXtreme Scale avec WebSphere Application Server», à la page 344.

Que faire ensuite

Arrêtez vos processus eXtreme Scale. Pour plus d'informations, voir «Arrêt de serveurs eXtreme Scale autonomes», à la page 339.

Démarrage du service de catalogue dans un environnement autonome

Vous devez démarrer le service de catalogue manuellement si vous utilisez un environnement WebSphere eXtreme Scale réparti qui n'est pas exécuté dans WebSphere Application Server.

Avant de commencer

Si vous utilisez WebSphere Application Server, le service de catalogue démarre automatiquement dans l'un des processus existants. Pour plus d'informations, voir Démarrage du processus du service de catalogue dans un environnement WebSphere Application Server.

Pourquoi et quand exécuter cette tâche

Le service de catalogue peut être exécuté dans un seul processus ou il peut inclure plusieurs serveurs de catalogues constituant un domaine de services de catalogue. Un domaine de serveurs de catalogues est indispensable en environnement de production si l'on veut bénéficier de la haute disponibilité. Pour plus d'informations sur la configuration d'un domaine de services de catalogue, voir dans le *Présentation du produit* les explications sur les domaines de services de catalogue. Le service de catalogue, qu'il soit placé dans une grille ou un dans un seul processus, est démarré à l'aide du script `startOgServer`. Vous pouvez également spécifier des paramètres supplémentaires au script pour associer l'ORB (Object Request Broker) à un hôte et un port spécifiques, spécifier le domaine ou activer la sécurité.

Lorsque vous appelez la commande de démarrage, utilisez le script `startOgServer.sh` sur les plateformes Unix ou `startOgServer.bat` sous Windows.

Procédure

- **Démarrage d'un processus de serveur de catalogues**

  Pour démarrer un serveur de catalogues, entrez les commandes suivantes à partir de la ligne de commande :

  1. Accédez au répertoire bin :

     ```
     cd objectgridRoot/bin
     ```

  2. Exécutez la commande startOgServer :

     ```
     startOgServer.bat|sh catalogServer
     ```

  Pour une liste de tous les paramètres de ligne de commande disponibles, voir «Script startOgServer», à la page 336. N'utilisez pas une seule machine virtuelle Java (JVM) pour exécuter le service de catalogue dans un environnement de production. Si le service de catalogue échoue, aucun nouveau client ne peut être acheminé vers l'instance eXtreme Scale déployée et aucune nouvelle instance ObjectGrid ne peut être ajoutée au domaine. Pour ces motifs, vous devez démarrer un ensemble de machines virtuelles Java pour pouvoir exécuter un domaine de services de catalogue.

- **Démarrage d'un domaine de services de catalogue multi-processus**

  Pour démarrer un ensemble de serveurs afin d'exécuter un service de catalogue, vous devez utiliser l'option **-catalogServiceEndPoints** sur le script `startOgServer`. Cet argument accepte une liste de points de contact de service de catalogue au format `nomServeur:nomHôte:portClient:portHomologue`. Chaque attribut est défini comme suit :

serverName
Spécifie un nom permettant d'identifier le processus que vous lancez.

hostName
Spécifie le nom d'hôte de l'ordinateur sur lequel le serveur est lancé.

clientPort
Indique le port utilisé pour les communications de la grille de catalogue homologue.

peerPort
Identique à haManagerPort. Indique le port utilisé pour les communications de la grille de catalogue homologue.

L'exemple suivant indique comment démarrer la première des trois machines virtuelles Java pour héberger un service de catalogue :

1. Accédez au répertoire bin :
   ```
   cd objectgridRoot/bin
   ```
2. Exécutez la commande startOgServer :
   ```
   startOgServer.bat|sh cs1 -catalogServiceEndpoints
   cs1:MyServer1.company.com:6601:6602,
   cs2:MyServer2.company.com:6601:6602,
   cs3:MyServer3.company.com:6601:6602
   ```

   Dans ce cas, le serveur `cs1` de l'hôte `MyServer1.company.com` est démarré. Ce nom de serveur est le premier argument transmis au script. Lors de l'initialisation du serveur `cs1`, les paramètres catalogServiceEndpoints sont examinés pour déterminer les ports alloués pour ce processus. La liste est également utilisée pour permettre au serveur `cs1` d'accepter les connexions des autres serveurs : `cs2` et `cs3`.
3. Pour démarrer les serveurs de catalogues restants de la liste, transmettez les arguments ci-après au script startOgServer. Démarrage du serveur `cs2` sur l'hôte `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
   ```

   Démarrage du serveur `cs3` sur `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
   ```

   Important : Démarrez tous les serveurs de catalogues en parallèle.

   Vous devez démarrer les serveurs de catalogues qui se trouvent dans une grille en parallèle, car chaque serveur s'interrompt pour attendre que les autres serveurs de catalogues aient rejoint le groupe central. Un serveur de catalogues configuré pour une grille ne démarre pas tant qu'il n'a pas identifié les autres membres du groupe. Le serveur de catalogues arrive à expiration si aucun autre serveur ne devient disponible.

- **Association de l'ORB à un hôte et un port spécifiques**

  En dehors des ports définis dans l'argument **catalogServiceEndpoints**, chaque service de catalogue utilise également un ORB (Object Request Broker) pour accepter les connexions des clients et des conteneurs. Par défaut, l'ORB écoute sur le port `2809` du système hôte local. Si vous souhaitez associer l'ORB à un

hôte et un port spécifiques sur la machine virtuelle Java d'un service de catalogue, utilisez les arguments **-listenerHost** et **-listenerPort**. L'exemple suivant explique comment démarrer le serveur de catalogues d'une machine virtuelle Java avec son ORB associé au port 7000 sur MyServer1.company.com :

```
startOgServer.sh catalogServer -listenerHost MyServer1.company.com
-listenerPort 7000
```

Chacun des conteneurs et clients eXtreme Scale doit être fourni avec des données de point de contact d'ORB de service de catalogue. Les clients n'ont besoin que d'un sous-ensemble de ces données, mais vous devez utiliser au moins deux points de contact pour la haute disponibilité.

- **Désignation du domaine**

  Un nom de domaine n'est pas requis lors du démarrage d'un service de catalogue. Le nom de domaine par défaut est `defaultDomain`. Pour affecter un nom à votre domaine, utilisez l'option **-domain**. L'exemple ci-après montre comment démarrer la machine virtuelle Java d'un service de catalogue avec le nom de domaine `myDomain`.

  ```
  startOgServer.sh catalogServer -domain myDomain
  ```

- **Démarrez un service de catalogue sécurisé**

  Vous pouvez démarrer un service de catalogue sécurisé en spécifiant les arguments suivants :

  -clusterSecurityFile et -clusterSecurityUrl
  : Ces arguments spécifient le fichier `objectGridSecurity.xml`, qui décrit les propriétés de sécurité communes à tous les serveurs (serveurs de catalogues et serveurs conteneurs). L'un des exemples de propriété est la configuration de l'authentificateur qui représente le registre d'utilisateurs et le mécanisme d'authentification.

  -serverProps
  : Indique le fichier de propriétés du serveur, qui contient les propriétés de sécurité spécifiques au serveur. Le nom de fichier spécifié pour cette propriété correspond à un chemin classique, tel que `c:/tmp/og/catalogserver.props`.

  Pour un exemple de démarrage d'un service de catalogue sécurisé, voir l'étape 2 du tutoriel sur la sécurité Java SE, dans le *Présentation du produit*.. Pour un exemple de fichier `objectGridSecurity.xml`, voir «Fichier XML du descripteur de sécurité», à la page 375.

Démarrage de processus de conteneur

Vous pouvez démarrer eXtreme Scale depuis la ligne de commande en utilisant une topologie de déploiement ou un fichier `server.properties`.

Pourquoi et quand exécuter cette tâche

Pour démarrer un processus de conteneur, vous avez besoin d'un fichier ObjectGrid XML. Ce fichier spécifie quels serveurs eXtreme Scale sont hébergés par le conteneur. Vérifiez que votre conteneur est équipé pour héberger chaque ObjectGrid dans le fichier XML que vous lui transmettez. Toutes les classes que ces ObjectGrids requièrent doivent se trouver dans le chemin d'accès aux classes pour le conteneur. Pour plus d'informations sur le fichier XMLObjectGrid, voir «Fichier `objectGrid.xsd`», à la page 160.

Procédure

- **Démarrez le processus de conteneur depuis la ligne de commande.**
  1. A partir de la ligne de commande, accédez au répertoire bin :
     ```
     cd objectgridRoot/bin
     ```
  2. Exécutez la commande suivante :
     ```
     startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

  Important : Sur le conteneur, l'option **-catalogServiceEndPoints** est utilisée pour référencer l'hôte et le port de la fonction ORB sur le service de catalogue. Le service de catalogue utilise les options **-listenerHost** et **-listenerPort** pour spécifier l'hôte et le port de la fonction ORB ou accepte la liaison par défaut. Lorsque vous démarrez un conteneur, utilisez l'option **-catalogServiceEndPoints** pour référencer les valeurs transmises aux options **-listenerHost** et **-listenerPort** sur le service de catalogue. Si les options **-listenerHost** et **-listenerPort** ne sont pas utilisées quand le service de catalogue est démarré, la fonction ORB est liée au port 2809 sur le système hôte local pour le service de catalogue. N'utilisez pas l'option **-catalogServiceEndPoints** pour référencer les hôtes et les ports transmis à l'option **-catalogServiceEndPoints** sur le service de catalogue. Sur le service de catalogue, l'option **-catalogServiceEndPoints** est utilisée pour spécifier les ports nécessaires pour une configuration de serveur statique.
  Ce processus est identifié par c0, le premier argument transmis au script. Utilisez le fichier `companyGrid.xml` pour démarrer le conteneur. Si votre fonction ORB de serveur de catalogue est exécutée sur un hôte différent que celui du conteneur ou qu'elle utilise un autre port que celui par défaut, vous devez utiliser l'argument **-catalogServiceEndPoints** pour vous connecter à la fonction ORB. Pour cet exemple, partez du principe qu'un unique service de catalogue est exécuté sur le port `2809` sur `MyServer1.company.com`

- **Démarrez le conteneur à l'aide d'une règle de déploiement.**

  Sans être nécessaire, une stratégie est recommandée pendant le démarrage du conteneur. La règle de déploiement est utilisée pour configurer le partitionnement et la réplication pour eXtreme Scale. La règle de déploiement peut également être utilisée pour influencer le comportement de positionnement. Comme l'exemple précédent ne fournit pas de fichier de règle de déploiement, l'exemple reçoit toutes les valeurs par défaut en ce qui concerne la réplication, le partitionnement et le positionnement. Donc, les mappes dans le CompanyGrid se trouvent dans un mapSet. Le mapSet n'est ni partitionné ni répliqué. Pour plus d'informations sur les fichiers de règle de déploiement, voir «Fichier XML du descripteur de la règle de déploiement», à la page 174. L'exemple suivant utilise le fichier `companyGridDpReplication.xml` pour démarrer une machine virtuelle Java de conteneur, c0 :

  1. A partir de la ligne de commande, accédez au répertoire bin :
     ```
     cd objectgridRoot/bin
     ```
  2. Exécutez la commande suivante :
     ```
     startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml
     -deploymentPolicyFile ../xml/companyGridDpReplication.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

  Remarque : Si vous avez des classes Java stockées dans un répertoire spécifique, au lieu d'altérer le script StartOgServer, vous pouvez lancer le serveur avec des arguments, comme suit : `-jvmArgs -cp C:\ . . . \DirectoryPOJOs\POJOs.jar` . Dans le fichier `companyGridDpReplication.xml`, un seul groupe de mappes contient toutes les mappes. Ce mapSet est divisé en 10 partitions. Chaque partition a une réplique synchrone et aucune réplique asynchrone. Tout

conteneur utilisant la règle de déploiement companyGridDpReplication.xml combinée au fichier XML ObjectGrid companyGrid.xml est également capable d'héberger des fragments de CompanyGrid. Démarrez une autre machine virtuelle Java de conteneur, c1 :

1. A partir de la ligne de commande, accédez au répertoire bin :
   ```
   cd objectgridRoot/bin
   ```
2. Exécutez la commande suivante :
   ```
   startOgServer.sh c1 -objectGridFile ../xml/companyGrid.xml
   -deploymentPolicyFile ../xml/companyGridDpReplication.xml
   -catalogServiceEndPoints MyServer1.company.com:2809
   ```

Chaque règle de déploiement contient au moins un élément objectgridDeployment. Quand un conteneur est démarré, il publie sa règle de déploiement sur le service de catalogue. Le service de catalogue examine chaque élément objectgridDeployment. Si l'attribut objectgridName correspond à l'attribut objectgridName d'un élément objectgridDeployment précédemment reçu, l'élément objectgridDeployment le plus récent est ignoré. Le premier élément objectgridDeployment reçu pour un attribut objectgridName spécifique est utilisé comme élément maître. Par exemple, partons du principe que la machine virtuelle Java c2 utilise une règle de déploiement qui divise le mapSet en nombre différent de partitions :

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>
```

Vous pouvez maintenant démarrer une troisième machine virtuelle Java, c2 :

1. A partir de la ligne de commande, accédez au répertoire bin :
   ```
   cd objectgridRoot/bin
   ```
2. Exécutez la commande suivante :
   ```
   startOgServer.sh c2 -objectGridFile ../xml/companyGrid.xml
   -deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml
   -catalogServiceEndPoints MyServer1.company.com:2809
   ```

Le conteneur sur la machine virtuelle Java c2 est démarré avec une règle de déploiement qui spécifie 5 partitions pour le mapSet1. Cependant, le service de catalogue contient déjà la copie maître de l'objectgridDeployment pour le CompanyGrid. Quand la machine virtuelle Java c0 a été démarrée, elle a spécifié que 10 partitions existent pour ce mapSet. Comme il s'agit du premier conteneur à démarrer et publier sa règle de déploiement, cette dernière devient la stratégie maître. En conséquence, toute valeur d'attribut objectgridDeployment égale à CompanyGrid dans une règle de déploiement suivante est ignorée.

- **Démarrez un conteneur à l'aide d'un fichier de propriétés de serveur.**

Vous pouvez utiliser un fichier de propriétés de serveur pour configurer la fonction de trace et la sécurité sur un conteneur. Exécutez les commandes suivantes pour démarrer un conteneur c3 avec un fichier de propriétés de serveur.

1. A partir de la ligne de commande, accédez au répertoire bin :

   ```
   cd objectgridRoot/bin
   ```

2. Exécutez la commande suivante :

   ```
   startOgServer.sh c3 -objectGridFile ../xml/companyGrid.xml
   -deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml
   -catalogServiceEndPoints MyServer1.company.com:2809
   -serverProps ../serverProps/server.properties
   ```

Voici un exemple de fichier server.properties :

```
server.properties
workingDirectory=
traceSpec=*=all=disabled
systemStreamToFileEnabled=true
enableMBeans=true
memoryThresholdPercentage=50
```

Il s'agit d'un fichier de propriétés de serveur de base dans lequel la sécurité n'est pas activée. Pour plus d'informations concernant le fichier server.properties, voir «Fichier de propriétés du serveur», à la page 185.

Script startOgServer

Le script startOgServer démarre les serveurs. Vous pouvez utiliser divers paramètres lorsque vous démarrez vos serveurs pour activer la trace, spécifiez des numéros de port, etc.

Rôle

Vous pouvez utiliser le script startOgServer pour démarrer les serveurs.

Emplacement

Le script startOgServer se trouve dans le répertoire bin du répertoire racine ; par exemple :

```
cd objectgridRoot/bin
```

Remarque : Si des classes Java sont stockées dans un répertoire spécifique, au lieu de modifier le script startOgServer, vous pouvez lancer le serveur avec des arguments, comme suit : -jvmArgs -cp C:\ . . . \DirectoryPOJOs\POJOs.jar

.

Syntaxe des serveurs de catalogues

Pour démarrer un serveur de catalogues :

Windows

```
startOgServer.bat <server> [options]
```

UNIX

```
startOgServer.sh <server>[options]
```

Pour démarrer un serveur de catalogues configuré par défaut, utilisez les commandes suivantes :

Windows

```
startOgServer.bat catalogServer
```

UNIX

```
startOgServer.sh catalogServer
```

Options de démarrage des serveurs de catalogues

Paramètres de démarrage d'un serveur de catalogues :

-catalogServiceEndPoints <server:serverHost:clientPort:peerPort,server:serverHost:clientPort:peerPort>
Sur le conteneur, l'option **-catalogServiceEndpoints** permet de référencer l'hôte et le port de l'ORB (Object Request Broker) sur le service de catalogue.

-clusterSecurityFile <fichier_xml_sécurité_cluster>
Indique l'emplacement du fichier XML du descripteur de sécurité.

-clusterSecurityUrl <URL du xml de la sécurité du cluster>

-domain <nom de domaine>

-listenerHost <nom d'hôte>
Valeur par défaut : `localhost`

Indique l'hôte du programme d'écoute pour les communications avec le protocole IIOP (Internet Inter-ORB Protocol).

-listenerPort <port>
Valeur par défaut : `2809`

Indique le port du programme d'écoute pour les communications avec le protocole IIOP.

-haManagerPort <port>
Valeur par défaut : Identique au port homologue. Si cette propriété n'est pas définie, le service de catalogue génère automatiquement un port disponible.

Indique le numéro de port utilisé par le gestionnaire de haute disponibilité.

-serverProps <fichier de propriétés du serveur>
Indique le fichier de propriétés du serveur qui contient les propriétés de sécurité spécifiques au serveur. Le nom de fichier spécifié pour cette propriété correspond simplement à un chemin classique, tel que `c:/tmp/og/catalogserver.props`.

-JMXServicePort <port>
Valeur par défaut : `1099`

Indique le numéro de port pour les communications avec JMX (Java Management Extensions).

-traceSpec <spécification de la trace>
Indique une chaîne qui spécifie la portée de la trace qui est activée au démarrage du serveur.

Exemple :

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <fichier de trace>
Indique le chemin d'un fichier dans lequel les informations de trace doivent être sauvegardées.

Exemple : `../logs/c4Trace.log`

-timeout <secondes>
Undique un nombre de secondes avant que le démarrage du serveur n'arrive à expiration.

-script <fichier script>
Crée un script personnalisé pour que les commandes que vous spécifiez démarrent les serveurs de catalogues ou les conteneurs, puis les paramètrent ou les éditent selon vos besoins.

-jvmArgs <arguments de la machine virtuelle Java>
Indique un ensemble d'arguments de machine virtuelle Java. Tout paramètre placé après le paramètre **-jvmArgs** est utilisé pour démarrer la machine virtuelle Java du serveur. Si le paramètre **-jvmArgs** est utilisé, vérifiez qu'il s'agit du dernier argument de script facultatif spécifié.

Exemple :-jvmArgs `-Xms256M -Xmx1G`

Syntaxe des serveurs conteneurs Windows

```
startOgServer.bat <serveur> -objectgridFile <fichier xml>
-deploymentPolicyFile <fichier xml> [options]
```

Windows

```
startOgServer.bat <serveur> -objectgridUrl <URL du xml>
-deploymentPolicyUrl <URL du xml> [options]
```

UNIX

```
startOgServer.sh <server> -objectgridFile <fichier xml>
-deploymentPolicyFile <fichier xml> [options]
```

UNIX

```
 startOgServer.sh <serveur> -objectgridUrl <URL du xml>
-deploymentPolicyUrl <URL du xml> [options]
```

Options des serveurs conteneurs

-catalogServiceEndPoints<nomHôte:port,nomHôte:port>
Valeur par défaut : localhost:2809

-deploymentPolicyFile <fichier xml de la règle de déploiement>
Indique le chemin d'accès au fichier de la règle de déploiement.

Exemple : `../xml/SimpleDP.xml`

-deploymentPolicyUrl <url de la règle de déploiement>

-objectgridFile <Fichier XML du descripteur d'ObjectGrid>
Indique le chemin d'accès au fichier du descripteur d'ObjectGrid.

-objectgridUrl <URL du descripteur d'ObjectGrid>

-listenerHost <nom d'hôte>
Valeur par défaut : `localhost`

Indique l'hôte du programme d'écoute pour les communications avec le protocole IIOP (Internet Inter-ORB Protocol).

-listenerPort <port>
Valeur par défaut : 2809

Indique le port du programme d'écoute pour les communications avec le protocole IIOP.

-serverProps <fichier de propriétés du serveur>
Indique le chemin d'accès au fichier de propriétés du serveur.

Exemple :../security/server.props

-zone <nom de zone>
Indique la zone à utiliser pour tous les conteneurs du serveur.

-traceSpec <spécification de la trace>
Indique une chaîne qui spécifie la portée de la trace qui est activée au démarrage du serveur.

Exemple :

- ObjectGrid=all=enabled
- ObjectGrid*=all=enabled

-traceFile <fichier de trace>
Indique le chemin d'un fichier dans lequel les informations de trace doivent être sauvegardées.

Exemple : ../logs/c4Trace.log

-timeout <secondes>
Undique un nombre de secondes avant que le démarrage du serveur n'arrive à expiration.

-script <fichier script>
Crée un script personnalisé pour que les commandes que vous spécifiez démarrent les serveurs de catalogues ou les conteneurs, puis les paramètrent ou les éditent selon vos besoins.

-jvmArgs <arguments de la machine virtuelle Java>
Indique un ensemble d'arguments de machine virtuelle Java. Tout paramètre placé après le paramètre **-jvmArgs** est utilisé pour démarrer la machine virtuelle Java du serveur. Si le paramètre **-jvmArgs** est utilisé, vérifiez qu'il s'agit du dernier argument de script facultatif spécifié.

Exemple :-jvmArgs -Xms256M -Xmx1G

Arrêt de serveurs eXtreme Scale autonomes

Vous pouvez utiliser le script stopOgServer pour arrêter les processus serveur.

Procédure

- Arrêtez les processus de conteneur eXtreme Scale.
  1. A partir de la ligne de commande, accédez au répertoire bin.
     cd *objectGridRoot*/bin
  2. Exécutez le script stopOgServer pour arrêter le serveur. L'exemple suivant arrête le serveur c0 :
     stopOgServer *c0* -catalogServiceEndPoints *MyServer1.company.com:2809*
     Utilisez le même script pour arrêter plusieurs serveurs avec une liste délimitée par des virgules, comme suit :
     stopOgServer c0,c1,c2 -catalogServiceEndPoints MyServer1.company.com:2809

Avertissement : L'option **-catalogServiceEndPoints** doit correspondre à l'option **-catalogServiceEndPoints** qui a servi à démarrer le conteneur. S'il n'a pas été fait usage de **-catalogServiceEndPoints** pour démarrer le conteneur, les valeurs par défaut seront probablement localhost ou le nom d'hôte et 2809 pour le port ORB de connexion au service de catalogue. Sinon, utilisez les valeurs passées à **-listenerHost** et à **-listenerPort** dans le service de catalogue. Si les options **-listenerHost** et **-listenerPort** ne sont pas utilisées quand le service de catalogue est démarré, la fonction ORB est liée au port 2809 sur le système hôte local pour le service de catalogue.

- Arrêtez les processus de service de catalogue eXtreme Scale.
  1. A partir de la ligne de commande, accédez au répertoire bin.
     ```
     cd objectGridRoot/bin
     ```
  2. Exécutez le script stopOgServer pour arrêter le serveur.
     ```
     stopOgServer.sh serveurCatalogue -catalogServiceEndPoints MyServer1.company.com:2809
     ```

  Avertissement : Lors de l'arrêt d'un service de catalogue, l'option **-catalogServiceEndPoints** sert à référencer dans le service de catalogue l'hôte et le port de l'ORB (Object Request Broker). Le service de catalogue utilise les options **-listenerHost** et **-listenerPort** pour spécifier l'hôte et le port pour la liaison ORB ou accepte la liaison par défaut. Si les options **-listenerHost** et **-listenerPort** ne sont pas utilisées quand le service de catalogue est démarré, la fonction ORB est liée au port 2809 sur le système hôte local pour le service de catalogue. L'option **-catalogServiceEndPoints** ne sera pas la même lors de l'arrêt d'un service de catalogue que lors de son démarrage.

  Démarrer un service de catalogue requiert des ports d'accès homologues et des ports d'accès clients si les ports par défaut n'ont pas été utilisés. En revanche, l'arrêt d'un service de catalogue ne requiert que le port de l'ORB.
- Activez la fonction de trace pour le processus d'arrêt du serveur. Si un conteneur ne parvient pas à s'arrêter, vous pouvez activer la fonction de trace pour vous aider dans le débogage du problème. Pour activer la fonction de trace au cours de l'arrêt d'un serveur, ajoutez les paramètres **-traceSpec** et **-traceFile** pour les commandes d'arrêt. Le paramètre **-traceSpec** spécifie le type de trace et le paramètre **-traceFile** spécifie le chemin d'accès et le nom du fichier à créer et à utiliser pour les données de trace.
  1. A partir de la ligne de commande, accédez au répertoire bin.
     ```
     cd objectGridRoot/bin
     ```
  2. Exécutez le script stopOgServer avec la fonction de trace activée.
     ```
     stopOgServer.sh c4 -catalogServiceEndPoints MyServer1.company.com:2809
      -traceFile ../logs/c4Trace.log -traceSpec ObjectGrid=all=enabled
     ```

     Une fois la trace obtenue, recherchez les erreurs relatives aux conflits de ports, aux classes manquantes, aux fichiers XML manquants ou incorrects, ou toute trace de pile. Suggestions de spécifications de trace au démarrage :
     - `ObjectGrid=all=enabled`
     - `ObjectGrid*=all=enabled`

     Pour connaître toutes les options de spécification de trace, voir «Options de trace», à la page 451.

Script stopOgServer

Le script stopOgServer arrête les serveurs.

Rôle

En spécifiant son nom et ses points de contact de service de catalogue, vous pouvez utiliser le script stopOgServer pour arrêter un serveur.

Emplacement

Le script stopOgServer se trouve dans le répertoire bin du répertoire racine ; par exemple :

```
cd objectgridRoot/bin
```

Syntaxe

Pour arrêter un serveur de catalogues :

Windows

```
stopOgServer.bat <serveur> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [options]
```

UNIX

```
stopOgServer.sh <serveur> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [options]
```

Pour arrêter un serveur conteneur ObjectGrid :

Windows

```
stopOgServer.bat <serveur> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [options]
```

UNIX

```
startOgServer.sh -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [options]
```

Options

-clientSecurityFile <fichier de propriétés du serveur>

-traceSpec <spécification de la trace>
Indique une chaîne qui spécifie la portée de la trace qui est activée au démarrage du serveur.

Exemple :

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <fichier de trace>
Indique le chemin d'un fichier dans lequel les informations de trace doivent être sauvegardées.

Exemple : `../logs/c4Trace.log`

-jvmArgs <arguments de la machine virtuelle Java>
Tout paramètre placé après l'option -jvmArgs est utilisé pour démarrer la machine virtuelle Java du serveur. Si l'option -jvmArgs est utilisée, vérifiez qu'il s'agit du dernier argument de script facultatif spécifié.

Démarrage et arrêt des serveurs sécurisés eXtreme Scale

Il est souvent nécessaire de sécuriser les serveurs pour votre environnement de déploiement. Pour cela, il faut configurer d'une certaine façon le démarrage et l'arrêt.

Démarrage d'un serveur de sécurité dans un environnement de système d'exploitation Java

Vous pouvez démarrer un service de catalogue ou des serveurs de conteneur comme décrit ci-dessous.

Démarrage d'un service de catalogue eXtreme Scale sécurisé

Le démarrage d'un processus de service de catalogue eXtreme Scale requiert deux fichiers de configuration de sécurité supplémentaires :

Fichier XML du descripteur de sécurité : le fichier XML du descripteur de sécurité décrit les propriétés de sécurité communes à tous les serveurs (y compris les serveurs de catalogue et les serveurs de conteneur). Un exemple de propriété est la configuration de l'authentificateur qui représente le registre utilisateur et le mécanisme d'authentification.

Fichier de propriétés du serveur. Le fichier de propriétés du serveur configure les propriétés de sécurité spécifiques au serveur.

Lorsque vous utilisez la commande StartOgServer.sh ou startOgServer.cat pour démarrer un processus de service de catalogue eXtreme Scale sécurisé, vous pouvez utiliser -clusterSecurityFile ou -clusterSecurityUrl pour définir le fichier XML du descripteur de sécurité en tant que type de fichier ou d'URL. Vous pouvez également utiliser –serverProps pour paramétrer le fichier de propriétés du serveur.

Démarrage d'un serveur de conteneur eXtreme Scale sécurisé

Le démarrage d'un serveur de conteneur eXtreme Scale sécurisé requiert un fichier de configuration de sécurité :

- **Fichier de propriétés du serveur :** ce fichier permet de configurer les propriétés de sécurité spécifiques au serveur. Pour plus d'informations, voir «Fichier de propriétés du serveur», à la page 185.

Lorsque vous utilisez la commande startOgServer.sh ou startOgServer.cat pour démarrer le serveur de conteneur eXtreme Scale, vous pouvez utiliser –serverProps pour paramétrer le fichier de propriétés du serveur. Il existe plusieurs façons de paramétrer le fichier de propriétés du serveur. Référez-vous à ce fichier pour plus d'informations.

Pour plus d'informations sur la façon d'utiliser la commande startOgServer.sh ou startOgServer.bat et ses options, référez-vous à «Script startOgServer», à la page 336.

Arrêt d'un serveur eXtreme Scale sécurisé

L'arrêt d'un processus de service de catalogue eXtreme Scale sécurisé ou d'un serveur de conteneur requiert un fichier de configuration de sécurité :

- **Fichier de propriétés du client :** ce fichier permet configurer les propriétés de sécurité du client. Ces propriétés de sécurité permettent à un client de se connecter à un serveur sécurisé. Pour plus d'informations, référez-vous à «Fichier de propriétés du client», à la page 205.

Lorsque vous utilisez la commande stopOgServer.sh ou stopOgServer.cat pour arrêter un processus de service de catalogue eXtreme Scale ou un serveur de conteneur, vous pouvez utiliser -clientSecurityFile pour définir les propriétés de sécurité du client.

Pour plus d'informations sur la façon d'utiliser la commande stopOgServer.sh ou stopOgServer.cat et ses options, voir «Script stopOgServer», à la page 340.

Démarrage d'un serveur de sécurité dans WebSphere Application Server

Le démarrage d'un serveur ObjectGrid sécurisé dans WebSphere Application Server est similaire au démarrage d'un serveur ObjectGrids non sécurisé, à la différence près que vous devez passer les fichiers de configuration de sécurité. Au lieu d'utiliser l'argument -[FICHIER_PROPRIETES] (par exemple -serverProps) dans la commande, comme c'est le cas dans un environnement Java SE, vous utilisez -D[FICHIER_PROPRIETES] dans les arguments génériques Java Virtual Machine (JVM).

Démarrage d'un service de catalogue sécurisé dans Websphere Application Server

Un serveur de catalogue contient deux niveaux d'informations de sécurité :

- `-Dobjectgrid.cluster.security.xml.url` : spécifie le fichier objectGridSecurity.xml décrivant les propriétés de sécurité communes à tous les serveurs (y compris les serveurs de catalogue et les serveurs de conteneur). Un exemple est la configuration de l'authentificateur, qui inclut le registre utilisateur et le mécanisme d'authentification. Le nom de fichier spécifié pour cette propriété doit être dans un format URL, tel que "file:///tmp/og/objectGridSecurity.xml".
- `-Dobjectgrid.server.props` : spécifie le fichier de propriétés du serveur, contenant les propriétés de sécurité spécifiques au serveur. Le nom de fichier spécifié pour cette propriété est au format de chemin de fichier simple, par exemple "c:/tmp/og/catalogserver.props". L'utilisation de -Dobjectgrid.security.server.props est dépréciée, mais vous pouvez continuer à l'utiliser pour la compatibilité rétroactive.

Pour démarrer un service de catalogue sécurisé dans WebSphere Application Server, suivez les instructions de la section "Intégré à WebSphere Application Server" dans «Sécurité de la grille», à la page 358.

Ensuite, définissez la propriété de sécurité dans l'argument JVM générique du processus.

```
-Dobjectgrid.cluster.security.xml.url=file:///tmp/og/
objectGridSecurity.xml-Dobjectgrid.server.props=/tmp/og/
catalog.server.props
```

Les étapes permettant d'ajouter les arguments JVM génériques sont les suivantes :

- Développez Administration système dans la vue des tâches gauche.

- Cliquez sur le processus WebSphere Application Server sur lequel est déployé le service de catalogue, par exemple "Gestionnaire de déploiement".
- Sur la page droite, développez "Gestion des processus et Java " sous "Infrastructure du serveur".
- Cliquez sur "Définition des processus".
- Cliquez sur "Java Virtual Machine" sous "Autres propriétés".
- Entrez les propriétés dans la zone de saisie Arguments JVM génériques.

Démarrage d'un serveur de conteneur sécurisé dans WebSphere Application Server

Lorsqu'il se connecte au serveur de catalogue, le serveur de conteneur obtient les paramètres de sécurité configurés dans objectGridSecurity.xml, tels que la configuration de l'authentificateur ou le délai d'attente de la session de connexion. De plus, les propriétés de sécurité spécifiques au serveur doivent être définies au niveau du serveur de conteneur, dans la propriété -Dobjectgrid.server.props.

Utilisez la propriété -Dobjectgrid.server.props property au lieu de la propriété -Dobjectgrid.security.server.propsproperty ; ce fichier de propriétés contient également des propriétés non liées à la sécurité. Le nom de fichier spécifié pour cette propriété est au format de chemin de fichier simple, tel que `c:/tmp/og/server.props`.

Suivez la même procédure que ci-dessus pour ajouter les propriétés de sécurité aux arguments JVM génériques.

Administration de WebSphere eXtreme Scale avec WebSphere Application Server

Vous pouvez exécuter les processus des services de catalogue et des serveurs conteneurs dans WebSphere Application Server. La procédure de configuration de ces serveurs est différente d'une configuration autonome. Je service de catalogue peut être automatiquement démarré dans des serveurs ou des gestionnaires de déploiement WebSphere Application Server. Le processus de conteneur démarre lorsqu'une application eXtreme Scale est déployée et démarrée dans l'environnement WebSphere Application Server.

Démarrage du processus du service de catalogue dans un environnement WebSphere Application Server

Les serveurs de catalogues de WebSphere eXtreme Scale peuvent être démarrés automatiquement dans un environnement WebSphere Application Server ou WebSphere Application Server Network Deployment. Vous pouvez configurer le service de catalogue pour qu'il puisse être exécuté dans tout processus de la cellule WebSphere. Un service de catalogue sans cluster, à un serveur, est acceptable pour les environnements de développement. Pour un environnement de production, vous devez utiliser un domaine de services de catalogue avec plusieurs serveurs de catalogue.

Avant de commencer

- Vous devez installer WebSphere Application Server et WebSphere eXtreme Scale.

  Si vous procédez à une installation graphique d'eXtreme Scale, vous aurez la possibilité d'étendre des profils existants, y compris celui du gestionnaire de déploiement. Vous pouvez également étendre des profils après l'installation en

vous servant de l'outil PMT de gestion des profils, dans sa version graphique ou à partir de la ligne de commande (manageprofiles.sh|bat). Les profils créés après l'installation du produit peuvent être étendus dans le cadre de la création de profils ou ultérieurement à l'aide de l'outil PMT. Il n'existe pas d'interface graphique de cet outil pour les installations 64 bits de WebSphere Application Server. Dans ce cas, utilisez le script manageprofiles à partir de la ligne de commande.

Pour plus d'informations, voir Chapitre 3, «Installation et déploiement de WebSphere eXtreme Scale», à la page 17.

- 7.1+ Définissez un domaine de services de catalogue. Pour plus d'informations, voir «Création de domaines de service de catalogue dans WebSphere Application Server», à la page 346.

Pourquoi et quand exécuter cette tâche

Le processus du service de catalogue peut être exécuté dans un processus WebSphere Application Server. Le lieu et le moment où s'exécute le service de catalogue dépend de l'édition de WebSphere Application Server que vous utilisez.

WebSphere Application Server de base
Le service de catalogue s'exécute dans le processus du serveur d'applications. Par défaut, son démarrage automatique *n'est pas* activé.

WebSphere Application Server Network Deployment
Le service de catalogue s'exécute automatiquement dans le processus du gestionnaire de déploiement, mais vous pouvez le configurer pour qu'il s'exécute dans un ou plusieurs processus d'agent de noeud ou de serveur d'applications.

Fonctionnalité dépréciée : Dans les précédentes versions, l'on définissait la propriété personnalisée catalog.services.cluster pour définir un groupe de serveurs de catalogue dans l'environnement WebSphere Application Server. Configurée dans une précédente version, cette propriété personnalisée reste en vigueur. Mais, si l'on crée une nouvelle configuration, l'on obtient les mêmes résultats en créant un domaine de services de catalogue dans la console d'administration de WebSphere Application Server. Pour plus d'informations, voir «Création de domaines de service de catalogue dans WebSphere Application Server», à la page 346.

Restriction : Les serveurs d'un environnement WebSphere Application Server ne peuvent pas posséder le même nom s'ils utilisent l'ORB pour communiquer entre eux. Vous pouvez contourner cette restriction en spécifiant la propriété système **-Dcom.ibm.websphere.orb.uniqueServerName=true** pour les processus de même nom. Voici deux exemples : Des serveurs nommés server1 sur chaque noeud sont utilisés comme grille de services de catalogue ou plusieurs agents de noeud sont utilisés pour constituer une grille de services de catalogue.

Procédure

- **Démarrage d'un serveur de catalogues dans WebSphere Application Server de base :**

  Si WebSphere eXtreme Scale est intégré dans un profil WebSphere Application Server non fédéré, le service de catalogue ne démarre pas automatiquement, excepté dans les deux cas suivants :

  – Une application configurée pour démarrer automatiquement un conteneur eXtreme Scale :le service de catalogue et le conteneur sont tous deux

démarrés automatiquement. Cette fonctionnalité simplifie le test des unités dans les environnements de développement comme Rational Application Developer car vous n'avez pas besoin de démarrer explicitement un service de catalogue. Pour plus d'informations, voir «Démarrage automatique de conteneurs eXtreme Scale dans les applications WebSphere Application Server», à la page 353.

  - Si un domaine de services de catalogue est défini.
- **Démarrage d'un service de catalogue dans WebSphere Application Server Network Deployment** Le service de catalogue démarre dans une cellule WebSphere Application Server Network Deployment dans les scénarios suivants :
  - Lorsque WebSphere eXtreme Scale est installé sur WebSphere Application Server Network Deployment, le service de catalogue démarre automatiquement dans le processus du gestionnaire de déploiement s'il est étendu pour permettre un développement rapide prêt à l'emploi.
  - Lorsque vous définissez un domaine de services de catalogue. Un domaine de services de catalogue est une collection de serveurs de catalogue définis. Ces serveurs de catalogue peuvent être démarrés sur des serveurs d'applications existants au sein de l'environnement WebSphere Application Server Network Deployment. Vous pouvez également définir des serveurs distants. Pour plus d'informations, voir «Création de domaines de service de catalogue dans WebSphere Application Server».

  Avertissement : En environnement de production, ne faites pas cohabiter les services de catalogue avec des serveurs conteneurs eXtreme Scale. Incluez le service de catalogue dans plusieurs processus d'agents de noeud ou sur un serveur d'applications qui n'héberge pas d'application eXtreme Scale.

  Après avoir défini un domaine de services de catalogue, vous devez redémarrer chaque serveur identifié. Lorsque vous redémarrez ces serveurs, le domaine de services de catalogue démarre lui aussi, automatiquement.

Création de domaines de service de catalogue dans WebSphere Application Server

Les domaines de service de catalogue définissent un groupe de serveurs de catalogues qui gèrent le positionnement des fragments et surveillent l'état des serveurs de conteneur dans votre grille de données.

Avant de commencer

- Installez WebSphere eXtreme Scale sur WebSphere Application Server. Pour plus d'informations, voir «Intégration de WebSphere eXtreme Scale ou WebSphere eXtreme Scale Client à WebSphere Application Server», à la page 22.

Pourquoi et quand exécuter cette tâche

En créant le domaine de service de catalogue, vous définissez une collection de serveurs de catalogues hautement disponibles.

Ces serveurs de catalogues peuvent être exécutés dans WebSphere Application Server dans un groupe central et une cellule unique. Le domaine de service de catalogue peut également définir un groupe de serveurs éloigné exécutés dans des processus Java SE différents ou d'autres cellules WebSphere Application Server. Lorsque vous définissez un domaine de service de catalogue qui place les serveurs de catalogues sur les serveurs d'applications de la cellule, les mécanismes des groupes centraux de WebSphere Application Server sont utilisés. Les membres d'un

même domaine de service de catalogue ne peuvent donc pas étendre les limites d'un groupe central et un domaine de service de catalogue ne peut donc pas étendre des cellules. Toutefois, WebSphere eXtreme Scale peut étendre des cellules en se connectant à un serveur de catalogues au delà des limites de ces cellules, comme un domaine de service de catalogue autonome ou un domaine de service de catalogue imbriqué dans une autre cellule.

Avertissement : Ne regroupez pas les services de catalogue et les serveurs de conteneur WebSphere eXtreme Scale dans un environnement de production. Incluez le service de catalogue dans plusieurs processus d'agent de noeud ou sur un serveurs d'applications qui n'héberge pas d'application WebSphere eXtreme Scale.

Vous pouvez également créer un domaine de service de catalogue si vous utilisez le mode autonome. Pour plus d'informations, voir «Démarrage du service de catalogue dans un environnement autonome», à la page 331.

Procédure

1. Créez le domaine de service de catalogue.
   a. Dans la console d'administration de WebSphere Application Server, cliquez sur **Administration du système → WebSphere eXtreme Scale → Domaines de service de catalogue → Nouveau**.
   b. Définissez un nom, une valeur par défaut et les données d'identification de l'authentification JMX de votre domaine de service de catalogue.
   c. Ajoutez des noeuds finaux de serveur de catalogues. Vous pouvez sélectionner des serveurs d'applications existants ou ajouter des serveurs éloignés qui exécutent un service de catalogue.
2. Testez la connexion avec votre domaine de service de catalogue.
   a. Dans la console d'administration de WebSphere Application Server, cliquez sur **Administration du système → WebSphere eXtreme Scale → Domaines de service de catalogue**.
   b. Sélectionnez le domaine de service de catalogue à tester et cliquez sur **Tester la connexion**. Lorsque vous cliquez sur ce bouton, tous les noeuds finaux de domaine de service de catalogue définis sont interrogés un par un et, si l'un d'eux est disponible, un message est renvoyé pour indiquer que la connexion avec le domaine de service de catalogue a abouti.
3. Si vous créez des serveurs de catalogue sur des serveurs d'applications existants, redémarrez les serveurs que vous avez sélectionnés. Le service de catalogue démarre automatiquement sur les serveurs sélectionnés.

Que faire ensuite

Vous pouvez également créer cette configuration à l'aide de l'outil wsadmin. Pour plus d'informations sur les commandes, voir «Tâches d'administration du domaine de service de catalogue».

Tâches d'administration du domaine de service de catalogue

Vous pouvez utiliser les langages de script Jacl ou Jython pour gérer les domaines de service de catalogue dans votre configuration WebSphere Application Server.

Conditions requises

WebSphere eXtreme Scale Client doit être installé dans votre environnement WebSphere Application Server.

Liste de toutes les tâches d'administration

Pour obtenir une liste de toutes les tâches d'administration associées aux domaines de catalogue de service, exécutez la commande suivante avec wsadmin :

```
wsadmin>$AdminTask help XSDomainManagement
```

Commandes

Les tâches d'administration des domaines de service de catalogue incluent les commandes suivantes :

- «createXSDomain»
- «deleteXSDomain», à la page 349
- «getDefaultXSDomain», à la page 349
- «listXSDomains», à la page 350
- «modifyXSDomain», à la page 350
- «testXSDomainConnection», à la page 351
- «testXSServerConnection», à la page 352

createXSDomain

La commande createXSDomain enregistre un nouveau domaine de service de catalogue.

Tableau 19. Arguments de la commande createXSDomain

Argument	Description
-name (obligatoire)	Spécifie le nom du domaine de service de catalogue à éditer.
-default	Indique si le domaine de service de catalogue correspond au domaine par défaut de la cellule. La valeur par défaut est `true`. (Booléen : `true` ou `false`)
-userID	Indique l'ID utilisateur du domaine de service de catalogue.
-password	Indique le mot de passe du domaine de service de catalogue.
-properties	Indique les propriétés personnalisées du domaine de service de catalogue.

Tableau 20. Arguments de l'étape defineDomainServers

Argument	Description
-name	Indique le nom du noeud final du service de catalogue.
-hostName	Indique le nom de l'hôte sur lequel se trouve le noeud final.
-endPoints	Indique les numéros de port du noeud final du domaine de service de catalogue.
-properties	Indique les propriétés personnalisées du noeud final du domaine de service de catalogue.

Valeur renvoyée :

Exemple de syntaxe en mode de traitement par lots

- Utilisation de Jacl :

```
$AdminTask createXSDomain {-name TestDomain -default true -userID
xsuser -password ***** -defineDomainServers
{{cs1 xhost1.ibm.com "" 5501,2809,1099} {cs2 xhost2.ibm.com "" 5501,2809,1099}}}
```

- Utilisation d'une chaîne Jython :

```
AdminTask.createXSDomain('[-name TestDomain -default true -userID
xsuser -password ***** -defineDomainServers
[[cs1 xhost1.ibm.com "" 5501,2809,1099] [cs2 xhost2.ibm.com "" 5501,2809,1099]]]')
```

Exemple de syntaxe en mode interactif

- Utilisation de Jacl :

```
$AdminTask createXSDomain {-interactive}
```

- Utilisation d'une chaîne Jython :

```
AdminTask.createXSDomain ('[-interactive]')
```

deleteXSDomain

La commande deleteXSDomain supprime un domaine de service de catalogue.

Paramètres requis :

-name
Spécifie le nom du domaine de service de catalogue à supprimer.

Valeur renvoyée :

Exemple de syntaxe en mode de traitement par lots

- Utilisation de Jacl :

```
$AdminTask deleteXSDomain {-name TestDomain }
```

- Utilisation d'une chaîne Jython :

```
AdminTask.deleteXSDomain('[-name TestDomain ]')
```

Exemple de syntaxe en mode interactif

- Utilisation de Jacl :

```
$AdminTask deleteXSDomain {-interactive}
```

- Utilisation d'une chaîne Jython :

```
AdminTask.deleteXSDomain ('[-interactive]')
```

getDefaultXSDomain

La commande getDefaultXSDomain renvoie le domaine de service de catalogue par défaut de la cellule.

Paramètres requis : Aucun

Valeur renvoyée : Nom du domaine de service de catalogue par défaut.

Exemple de syntaxe en mode de traitement par lots

- Utilisation de Jacl :

```
$AdminTask getDefaultXSDomain
```

- Utilisation d'une chaîne Jython :
  ```
  AdminTask.getDefaultXSDomain
  ```

Exemple de syntaxe en mode interactif

- Utilisation de Jacl :
  ```
  $AdminTask getDefaultXSDomain {-interactive}
  ```
- Utilisation d'une chaîne Jython :
  ```
  AdminTask.getDefaultXSDomain ('[-interactive]')
  ```

listXSDomains

La commande listXSDomains renvoie une liste des domaines de service de catalogue existants.

Paramètres requis : Aucun

Valeur renvoyée : Liste de tous les domaines de service de catalogue de la cellule.

Exemple de syntaxe en mode de traitement par lots

- Utilisation de Jacl :
  ```
  $AdminTask listXSDomains
  ```
- Utilisation d'une chaîne Jython :
  ```
  AdminTask.listXSDomains
  ```

Exemple de syntaxe en mode interactif

- Utilisation de Jacl :
  ```
  $AdminTask listXSDomains {-interactive}
  ```
- Utilisation d'une chaîne Jython :
  ```
  AdminTask.listXSDomains ('[-interactive]')
  ```

modifyXSDomain

La commande modifyXSDomain modifie un domaine de service de catalogue existant.

Tableau 21. Arguments de la commande modifyXSDomain

Argument	Description
-name (obligatoire)	Spécifie le nom du domaine de service de catalogue à éditer.
-default	Si la valeur est `true`, indique que le domaine de service de catalogue sélectionné correspond au domaine par défaut de la cellule. (Booléen)
-userID	Indique l'ID utilisateur du domaine de service de catalogue.
-password	Indique le mot de passe du domaine de service de catalogue.
-properties	Indique les propriétés personnalisées du domaine de service de catalogue.

Tableau 22. Arguments de l'étape modifyEndpoints

Argument	Description
-name	Indique le nom du noeud final du service de catalogue.
-hostName	Indique le nom de l'hôte sur lequel se trouve le noeud final.
-endPoints	Indique les numéros de port du noeud final du domaine de service de catalogue.

Tableau 23. Arguments de l'étape addEndpoints

Argument	Description
-name	Indique le nom du noeud final du service de catalogue.
-hostName	Indique le nom de l'hôte sur lequel se trouve le noeud final.
-endPoints	Indique les numéros de port du noeud final du domaine de service de catalogue.
-properties	Indique les propriétés personnalisées du noeud final du domaine de service de catalogue.

Tableau 24. Arguments de l'étape removeEndpoints

Argument	Description
-name	Spécifie le nom du noeud final de service de catalogue à supprimer.

Valeur renvoyée :

Exemple de syntaxe en mode de traitement par lots

- Utilisation de Jacl :

```
$AdminTask modifyXSDomain {-name TestDomain -userID newuser -password *****
-default false -modifyEndpoints {{cs1 xhost1.ibm.com "" 5502,2809,1099}}
-addEndpoints {{cs3 xhost3.ibm.com "" 5501,2809,1099}}
-removeEndpoints {{cs2}}}
```

- Utilisation d'une chaîne Jython :

```
AdminTask.modifyXSDomain('[-name TestDomain -userID newuser -password *****
-default false -modifyEndpoints [[cs1 xhost1.ibm.com "" 5502,2809,1099]]
-addEndpoints [[cs3 xhost3.ibm.com "" 5501,2809,1099]] -removeEndpoints
[[cs2]]]')
```

Exemple de syntaxe en mode interactif

- Utilisation de Jacl :

```
$AdminTask modifyXSDomain {-interactive}
```

- Utilisation d'une chaîne Jython :

```
AdminTask.modifyXSDomain ('[-interactive]')
```

testXSDomainConnection

La commande testXSDomainConnection teste la connexion à un domaine de service de catalogue.

Paramètres requis :

-name
Spécifie le nom du domaine de service de catalogue avec lequel la connexion doit être testée.

Paramètres facultatifs

-timeout
Indique la durée d'attente maximale, en secondes, de la connexion.

Valeur renvoyée : Si une connexion peut être établie, renvoie `true` ; sinon, des informations sur l'erreur de connexion sont renvoyées.

Exemple de syntaxe en mode de traitement par lots

- Utilisation de Jacl :
  ```
  $Admintask testXSDomainConnection
  ```
- Utilisation d'une chaîne Jython :
  ```
  AdminTask.testXSDomainConnection
  ```

Exemple de syntaxe en mode interactif

- Utilisation de Jacl :
  ```
  $AdminTask testXSDomainConnection {-interactive}
  ```
- Utilisation d'une chaîne Jython :
  ```
  AdminTask.testXSDomainConnection ('[-interactive]')
  ```

testXSServerConnection

La commande testXSServerConnection teste la connexion à un serveur de catalogue. Cette commande fonctionne pour les serveurs autonomes et les serveurs qui font partie d'un domaine de service de catalogue.

Paramètres requis :

hôte
Indique l'hôte sur lequel se trouve le serveur de catalogue.

listenerPort
Indique le port du programme d'écoute du serveur de catalogue.

Paramètres facultatifs

délai d'attente
Indique la durée d'attente maximale, en secondes, d'une connexion avec le serveur de catalogues.

Valeur renvoyée:

Exemple de syntaxe en mode de traitement par lots

- Utilisation de Jacl :
  ```
  $Admintask testXSServerConnection {-host xhost1.ibm.com -listenerPort 2809}
  ```
- Utilisation d'une chaîne Jython :
  ```
  AdminTask.testXSServerConnection('[-host xshost3.ibm.com -listenerPort 2809]')
  ```

Exemple de syntaxe en mode interactif

- Utilisation de Jacl :

```
$AdminTask testXSServerConnection {-interactive}
```

- Utilisation d'une chaîne Jython :

```
AdminTask.testXSServerConnection ('[-interactive]')
```

Démarrage automatique de conteneurs eXtreme Scale dans les applications WebSphere Application Server

Les processus de conteneur dans un environnement WebSphere Application Server démarrent automatiquement quand un module ayant les fichiers XML eXtreme Scale inclus démarre.

Avant de commencer

WebSphere Application Server et WebSphere eXtreme Scale doivent être installés, et vous devez être capable d'accéder à la console d'administration de WebSphere Application Server.

Pourquoi et quand exécuter cette tâche

Les applications Java Platform, Enterprise Edition ont des règles de chargeur de classe complexes qui compliquent grandement les classes de chargement lors de l'utilisation d'un WebSphere eXtreme Scale partagé au sein d'un serveur Java EE. Une application Java EE se compose typiquement d'un unique fichier d'archive d'entreprise (EAR) avec au moins un module Enterprise JavaBeans™ (EJB) ou d'archive Web déployé dedans.

WebSphere eXtreme Scale surveille le démarrage de chaque module et recherche des fichiers XML eXtreme Scale. Si WebSphere eXtreme Scale détecte qu'un module démarre avec les fichiers XML, il enregistre le serveur d'application en tant que conteneur eXtreme Scale container machine virtuelle Java (JVM) avec le service de catalogue. En enregistrant les conteneurs avec le service de catalogue, la même application peut être déployée dans différentes grilles et toujours être traitée comme une seule grille par le service de catalogue. Le service de catalogue ne prend pas en compte les cellules, les grilles ou les grilles dynamiques. Soit c'est une JVM de conteneur eXtreme Scale soit ce n'en est pas une. Une simple grille logique peut s'étendre sur plusieurs cellules WebSphere Extended Deployment si nécessaire.

Procédure

1. Créez un package de votre fichier EAR pour avoir les modules incluant les fichiers XML eXtreme Scale dans le dossier META-INF. WebSphere eXtreme Scale détecte la présence des fichiers `objectGrid.xml` et `objectGridDeployment.xml` dans le dossier META-INF des modules EJB et WEB quand ils démarrent. Si un seul fichier `objectGrid.xml` est trouvé, la machine virtuelle Java estime être le client, sinon, nous considéons que cette machine virtuelle Java fait office de conteneur pour la grille définie dans le fichier `objectGridDeployment.xml`.

   Vous devez utiliser les noms corrects pour ces fichiers XML. Les noms de fichiers sont sensibles à la casse. Si les fichiers sont absents, le conteneur ne démarre pas. Vous pouvez vérifier si le fichier `systemout.log` contient des messages indiquant que des fragments sont placés. Un module EJB ou d'archive Web utilisant eXtreme Scale doit avoir des fichiers XML eXtreme Scale dans son répertoire `META-INF`.

   Les fichiers XML eXtreme Scale incluent :

- Un fichier `objectGrid.xml`, fréquemment décrit comme un fichier XML descripteur eXtreme Scale.
- Un fichier `objectGridDeployment.xml`, fréquemment décrit comme un fichier XML descripteur de déploiement.
- Un fichier `entity.xml`, si des entités sont utilisées (facultatif). Le nom du fichier `entity.xml` doit correspondre au nom spécifié dans le fichier `objectGrid.xml`.

La phase d'exécution de eXtreme Scale détecte ces fichiers, puis contacte le service de catalogue pour l'informer qu'un autre conteneur est disponible pour héberger les fragments pour ce eXtreme Scale.

Conseil : Si vous prévoyez d'essayer une application avec des entités à l'aide d'un seul serveur eXtreme Scale, donnez la valeur `0` à minSyncReplicas dans le fichier XML du descripteur de déploiement. Sinon, vous risquez de voir l'un des messages suivants dans le fichier `SystemOut.log` car le positionnement ne pourra se produire tant qu'un autre serveur n'a pas démarré pour satisfaire à la règle minSyncReplica :

```
CWPRJ1005E: Erreur lors de la résolution de l'association d'entités.
Entité=nom_entité,association=nom_association.

CWOBJ3013E: Le référentiel EntityMetadata n'est pas disponible. Le seuil du

délai d'attente a été atteint lors de la tentative d'inscription de
l'entité : nom_entité.
```

2. Déployez et démarrez votre application.

   Le conteneur démarre automatiquement quand le module est démarré. Le service de catalogue commence à placer les serveurs principaux et secondaires de partition (fragments) dès que possible. Ce positionnement se produit immédiatement, à moins que vous ne définissiez un attribut numInitialContainers dans le fichier `objectGridDeployment.xml`. Si vous définissez cet attribut, le positionnement démarre quand ce nombre de conteneurs ont démarré.

Que faire ensuite

Les applications qui se trouvent dans la même cellule que les conteneurs, peuvent se connecter à ces grilles en utilisant une méthode ObjectGridManager.connect(null, null), puis appeler la méthode getObjectGrid(ccc, "nom de l'objet grille"). Les méthodes connect et getObjectGrid peuvent être bloquées jusqu'au positionnement des fragments par les conteneurs, mais ce blocage représente un problème uniquement quand la grille démarre.

Chargeurs de classe

Tout plug-in ou objet stocké dans un eXtreme Scale est chargé sur un certain chargeur de classe. Deux modules EJB dans un même fichier d'archive d'entreprise peuvent inclure ces objets. Ces objets sont les mêmes mais sont chargés à l'aide de ClassLoaders différents. Si l'application A stocke un objet Personne dans une mappe eXtreme Scale qui est locale pour le serveur, l'application B reçoit une ClassCastException si elle essaie de lire cet objet. Cette exception se produit car l'application B a chargé l'objet Personne sur un chargeur de classe différent.

Une manière de résoudre ce problème consiste à faire en sorte qu'un module racine contienne les plug-in et les objets nécessaires qui sont stockés dans le eXtreme Scale. Chaque module utilisant eXtreme Scale doit référencer ce module pour ses classes. Une autre possibilité consiste à placer ces objets partagés dans un fichier

jar qui se trouve sur un chargeur de classe commun partagé par les modules et les applications. Les objets peuvent également être placés dans les classes WebSphere ou le répertoire `lib/ext`, mais cela complique le déploiement et n'est donc pas recommandé.

Les modules EJB dans un fichier d'archive d'entreprise partagent généralement le même ClassLoader et ne sont pas affectés par ce problème. Chaque module de fichier d'archive Web possède son propre ClassLoader et est affecté par ce problème.

Connexion à une grille uniquement en tant que client

Si la propriété catalog.services.cluster est définie dans les propriétés personnalisées de la cellule, du nœud ou du serveur, tout module du fichier d'archive d'entreprise peut appeler la méthode ObjectGridManager.connect (ServerFactory.getServerProperties().getCatalogServiceBootstrap(), null, null) pour obtenir un ClientClusterContext et appeler la méthode ObjectGridManager.getObjectGrid(ccc, "grid name") pour obtenir une référence vers le eXtreme Scale. Si des objets d'application sont stockés dans les mappes, vérifiez qu'ils sont présents dans un ClassLoader commun.

Les clients Java Platform, Standard Edition ou les clients en dehors de la cellule peuvent se connecter à l'aide du port IIOP d'amorçage du service de catalogue (le gestionnaire de déploiement, par défaut dans WebSphere) pour obtenir un ClientClusterContext et un eXtreme Scale nommé de la manière habituelle.

Gestionnaire d'entités

Le gestionnaire d'entités est utile car les blocs de données sont stockés dans les mappes, non dans les objets application, et il y a donc moins de risques de problèmes de chargeur de classe. Les plug-in, en revanche, peuvent présenter un problème. Notez également qu'un fichier XML descripteur eXtreme Scale de substitution de client est toujours requis lors de la connexion à un eXtreme Scale qui possède des entités définies : ObjectGridManager.connect("host:port[,host:port], null, objectGridOverride) ou ObjectGridManager.connect(null, objectGridOverride).

Administration par programmation à l'aide de beans gérés (MBeans)

Vous pouvez utiliser plusieurs types de beans gérés JMX (Java Management Extensions) différents pour administrer et surveiller les déploiements. Chaque bean géré fait référence à une entité spécifique, une mappe, une grille de données, un serveur ou un service.

Interfaces MBean JMX et WebSphere eXtreme Scale

Chaque bean géré contient des méthodes get qui représentent des valeurs d'attribut. Ces méthodes get ne peuvent pas être appelées directement à partir de votre programme. La spécification JMX traite les attributs différemment des opérations. Vous pouvez afficher les attributs à l'aide de la console JMX d'un fournisseur et effectuer des opérations dans votre programme ou à l'aide de la console JMX d'un fournisseur.

Package com.ibm.websphere.objectgrid.management

Vous trouverez dans la documentation de l'API une présentation d'ensemble et des spécifications détaillées pour la programmation de tous les beans gérés

utilisables :Package com.ibm.websphere.objectgrid.management.

Accès aux beans gérés à l'aide de l'outil wsadmin

Vous pouvez utiliser l'utilitaire wsadmin fourni dans WebSphere Application Server pour accéder aux informations sur les beans gérés.

Exécutez l'outil wsadmin depuis le répertoire bin de votre installation WebSphere Application Server. L'exemple suivant restaure une vue de la position actuelle du fragment dans un logiciel eXtreme Scale dynamique. Vous pouvez exécuter wsadmin depuis n'importe quelle installation où eXtreme Scale est en cours d'exécution. Vous n'avez pas besoin de l'exécuter sur le service de catalogue.

```
$ 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>
```

Chapitre 8. Sécurisation de l'environnement de déploiement

WebSphere eXtreme Scale peut sécuriser l'accès aux données, y compris en permettant de s'intégrer à des fournisseurs externes de sécurité. La sécurité recouvre plusieurs aspects : authentification, autorisation, sécurité des transports, sécurité des grilles, sécurité locale et sécurité de JMX (Mbean).

Activation de la sécurité locale

WebSphere eXtreme Scale fournit plusieurs points de contact de sécurité permettant d'intégrer les mécanismes personnalisés. Dans le modèle de programmation local, la principale fonction de sécurité est l'autorisation, qui n'est associée à aucune prise en charge de l'authentification. Vous devez vous authentifier indépendamment de l'authentification WebSphere Application Server existante. Toutefois, des plug-in permettant d'obtenir et de valider des objets Subject sont fournis.

Les sections suivantes décrivent deux manières différentes d'activer la sécurité locale :

Vous pouvez utiliser le fichier XML ObjectGrid pour définir une grille d'objets et activer la sécurité pour cette grille. Le fichier `secure-objectgrid-definition.xml` utilisé dans l'exemple d'application d'entreprise ObjectGridSample est présenté dans l'exemple suivant. Pour activer la sécurité, associez l'attribut securityEnabled à la valeur `true`.

```
<objectGrids>
    <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
        authorizationMechanism="AUTHORIZATION_MECHANISM_JAAS">
    ...
</objectGrids>
```

Vous pouvez également activer la sécurité à l'aide d'un programme. Pour créer une grille d'objets à l'aide de la méthode ObjectGrid.setSecurityEnabled, appelez la méthode suivante dans l'interface ObjectGrid :

```
/**
 * L'on active la sécurité de la grille d'objets
 */
void setSecurityEnabled();
```

Pour plus d'informations, consultez la documentation relative aux API..

Authentification de grille

Vous pouvez utiliser le plug-in du gestionnaire de jetons de sécurité pour activer l'authentification serveur à serveur, ce qui signifie d'implémenter l'interface SecureTokenManager.

La méthode generateToken(Object) prend un objet protect, puis génère un jeton pouvant être compris par les autres. La méthode verifyTokens(byte[]) procède en sens contraire : elle reconvertit le jeton en objet d'origine.

Une implémentation SecureTokenManager simple utilise un algorithme de codage de base, tel qu'un algorithme XOR, pour coder l'objet dans un formulaire sérialisé et utiliser l'algorithme de décodage correspondant pour décoder le jeton. Cette implémentation n'est pas sécurisée et peut être facilement interrompue.

Implémentation par défaut de WebSphere eXtreme Scale

WebSphere eXtreme Scale met immédiatement à disposition une implémentation de cette interface. Cette implémentation par défaut utilise une paire de clés pour signer et vérifier la signature et une clé confidentielle pour chiffrer le contenu. Chaque serveur comporte un fichier de clés de type JCKES contenant la paire de clés, une clé privée et une clé publique ainsi qu'une clé confidentielle. Pour stocker les clés confidentielles, le fichier de clés doit être de type JCKES. En effet, ces clés servent à chiffrer et signer ou vérifier la chaîne secrète côté expéditeur. De plus, le jeton est associé à un délai d'expiration. Côté récepteur, les données sont vérifiées, déchiffrées et comparées à la chaîne secrète du récepteur. Des protocoles de communication SSL (Secure Sockets Layer) ne sont pas requis entre une paire de serveurs pour l'authentification car les clés privées et les clés publiques ont la même finalité. Toutefois, si la communication du serveur n'est pas chiffrée, les données peuvent être dérobées à la seule vue de la communication. Le jeton venant à expiration, la menace d'attaque par relecture est minimisée. Ce risque est considérablement réduit si tous les serveurs sont déployés derrière un pare-feu.

L'inconvénient de cette approche : les administrateurs de WebSphere eXtreme Scale doivent générer des clés et les transporter vers tous les serveurs, ce qui peut provoquer des failles de sécurité lors du transport.

Sécurité de la grille

Grâce à la fonction de sécurité de la grille de WebSphere eXtreme Scale, un serveur rejoignant la grille dispose de données d'identification correctes, tandis qu'un serveur malveillant n'y est pas autorisé. Cette fonction utilise un mécanisme de chaîne de secret partagé.

Tous les serveurs WebSphere eXtreme Scale, y compris les serveurs de catalogues, choisissent une chaîne de secret partagé commune. Lorsqu'un serveur rejoint la grille, il doit présenter la chaîne confidentielle. Si cette chaîne correspond à celle du serveur président ou du serveur de catalogues, le serveur est accepté. Dans le cas contraire, la demande de jointure est rejetée.

L'envoi d'un texte en clair n'est pas sécurisé. L'infrastructure de sécurité WebSphere eXtreme Scale propose un plug-in de gestionnaire de jetons sécurisé permettant au serveur de sécuriser la valeur confidentielle avant son envoi. Vous devez choisir le mode d'implémentation de cette opération. WebSphere eXtreme Scale propose une implémentation prête à l'emploi : l'opération de sécurité est implémentée pour chiffrer et signer la valeur confidentielle.

La chaîne confidentielle est définie dans le fichier `server.properties`. Reportez-vous à la rubrique «Fichier de propriétés du serveur», à la page 185 pour plus d'informations sur la propriété authenticationSecret.

Plug-in SecureTokenManager

Le plug-in de gestionnaire de jetons sécurisé est représenté par l'interface com.ibm.websphere.objectgrid.security.plugins.SecureTokenManager.

Pour plus d'informations sur ce plug-in, consultez la documentation relative à l'API SecureTokenManager.

La méthode generateToken(Object) prend un objet, puis génère un jeton ne pouvant être compris par les autres. La méthode verifyTokens(byte[]) suit le processus inverse : la méthode reconvertit le jeton à son format d'origine.

Une implémentation SecureTokenManager simple utilise un algorithme de codage simple, par exemple un algorithme OU exclusif (XOR), pour coder l'objet au format sérialisé, puis utilise l'algorithme de décodage pour décoder le jeton. Cette implémentation n'est pas sécurisée.

WebSphere eXtreme Scale propose une implémentation disponible immédiatement pour cette interface.

L'implémentation par défaut utilise une paire de clés pour signer et vérifier la signature et utilise une clé confidentielle pour en chiffrer le contenu. Chaque serveur a un fichier de clés de type JCKES pour stocker cette paire (une clé privée et une clé publique), ainsi qu'une clé confidentielle. Le fichier de clés doit être de type JCKES pour pouvoir stocker les clés confidentielles.

Ces clés sont utilisées pour chiffrer et signer ou vérifier la chaîne secrète côté envoi. Le jeton est associé à un délai d'expiration. Côté réception, les données sont vérifiées, déchiffrées et comparées à la chaîne secrète du récepteur. Les protocoles de communication SSL (Secure Sockets Layer) ne sont pas obligatoires pour l'authentification entre une paire de serveurs car les clés privées et les clés publiques ont les mêmes fonctions. Toutefois, si la communication avec les serveurs n'est pas chiffrée, les données peuvent être volées en regardant la communication. Le jeton expirant bientôt, la menace pesant sur les attaques de type replay est minime. Cette possibilité décroît même de manière significative lorsque tous les serveurs sont déployés derrière un pare-feu.

L'inconvénient de cette approche est que les administrateurs WebSphere eXtreme Scale doivent générer des clés et en assurer le transport vers tous les serveurs, au cours duquel la sécurité risque d'être enfreinte.

Exemples de scripts permettant de créer les propriétés du gestionnaire de jetons par défaut

Comme indiqué dans la précédente section, vous pouvez créer un fichier de clés contenant une paire de clés pour signer et vérifier la signature, ainsi qu'une clé confidentielle pour chiffrer le contenu.

Vous pouvez par exemple utiliser la commande de l'outil de clé JDK 6 pour créer les clés, comme ci-dessous :

```
keytool -genkeypair -alias keypair1 -keystore key1.jck -storetype JCEKS -keyalg
rsa -dname "CN=sample.ibm.com, OU=WebSphere eXtreme Scale" -storepass key111 -keypass
keypair1 -validity 10000
```

```
keytool -genseckey -alias seckey1 -keystore key1.jck -storetype JCEKS -keyalg
DES  -storepass key111 -keypass seckey1 -validity 1000
```

Ces deux commandes créent une paire de clés "keypair1" et une clé confidentielle "seckey1". Vous pouvez alors configurer les éléments suivants dans le fichier de propriétés du serveur :

```
secureTokenKeyStore=key1.jck
secureTokenKeyStorePassword=key111
secureTokenKeyStoreType=JCEKS
secureTokenKeyPairAlias=keypair1
secureTokenKeyPairPassword=keypair1
```

```
secureTokenSecretKeyAlias=seckey1
secureTokenSecretKeyPassword=seckey1
secureTokenCipherAlgorithm=DES
secureTokenSignAlgorithm=RSA
```

Configuration

Consultez la rubrique Propriétés du serveur pour plus d'informations sur les propriétés utilisées pour configurer le gestionnaire de jetons sécurisé.

Authentification du client d'application

L'authentification du client d'application consiste à activer la sécurité client-serveur et l'authentification des données d'identification et à configurer un authentificateur et un générateur de données d'identification.

Activation de la sécurité client-serveur

Vous devez activer la sécurité sur le client et sur le serveur pour pouvoir vous authentifier auprès de la grille d'objets.

Activation de la sécurité du client

WebSphere eXtreme Scale fournit un exemple de fichier de propriétés, le fichier `sampleClient.properties`, situé dans le répertoire *WAS_HOME*`/optionalLibraries/ObjectGrid/properties` pour une installation WebSphere Extended Deployment ou dans le répertoire `/ObjectGrid/properties` d'une installation contenant plusieurs types de serveurs. Vous pouvez modifier ce fichier en y entrant les valeurs de votre choix. Associez la propriété securityEnabled du fichier `objectgridClient.properties` à la valeur `true`. La propriété securityEnabled indique si la sécurité est activée. Lorsqu'un client se connecte à un serveur, les valeurs du côté client et du côté serveur doivent toutes deux être égales à `true` ou à `false`. Par exemple, si la sécurité du serveur connecté est activée, la valeur de la propriété doit être associée à `true` du côté client pour que le client puisse se connecter au serveur.

L'interface com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration représente le fichier `security.ogclient.props`. Vous pouvez utiliser l'API publique com.ibm.websphere.objectgrid.security.config.ClientSecurityConfigurationFactory pour créer une instance de cette interface avec les valeurs par défaut ou vous pouvez créer une instance en transmettant le fichier des propriétés de sécurité du client ObjectGrid. Le fichier `security.ogclient.props` contient d'autres propriétés. Pour plus de détails, voir la documentation de l'API ClientSecurityConfiguration et celle de l'API ClientSecurityConfigurationFactory.

Activation de la sécurité du serveur

Pour activer la sécurité côté serveur, vous pouvez donner la valeur `true` à la propriété **securityEnabled** dans le fichier `security.xml`. Un fichier XML de descripteur de sécurité vous permettra de spécifier la configuration de la sécurité de la grille en isolant celle-ci de tous les éléments de configuration n'ayant pas trait à la sécurité.

Activation de l'authentification des données d'identification

Une fois que le client eXtreme Scale l'a récupéré à l'aide de l'objet CredentialGenerator, l'objet Credential est envoyé au serveur eXtreme Scale avec la demande du client. Le serveur authentifie l'objet Credential avant de traiter la demande. Si l'authentification réussit, un objet Subject est renvoyé pour représenter cet objet Credential. Cet objet Subject est alors utilisé pour autoriser la demande.

Définissez la propriété **credentialAuthentication** dans les fichiers de propriétés du client et du serveur afin d'activer l'authentification des données d'identification. Pour plus d'informations, voir «Fichier de propriétés du client», à la page 205 et «Fichier de propriétés du serveur», à la page 185.

Le tableau suivant présente les mécanismes d'authentification à utiliser selon les paramètres.

Tableau 25. Authentification des données d'identification dans les paramètres du client et du serveur

Authentification des données d'identification du client	Authentification des données d'identification du serveur	Résultat
Non	Jamais	Désactivée
Non	Prise en charge	Désactivée
Non	Obligatoire	Cas d'erreur
Prise en charge	Jamais	Désactivée
Prise en charge	Prise en charge	Activé
Prise en charge	Obligatoire	Activé
Obligatoire	Jamais	Cas d'erreur
Obligatoire	Prise en charge	Activé
Obligatoire	Obligatoire	Activé

Configuration d'un authentificateur

Le serveur eXtreme Scale utilise le plug-in Authenticator pour authentifier l'objet Credential. Une implémentation de l'interface Authenticator obtient l'objet Credential qu'elle authentifie ensuite auprès d'un registre d'utilisateurs, un serveur LDAP, par exemple, et ainsi de suite. eXtreme Scale ne fournit aucune configuration de registre. La connexion à un registre d'utilisateurs et l'authentification auprès de celui-ci doivent être implémentées dans ce plug-in.

Par exemple, une implémentation d'Authenticator extrait l'ID utilisateur et le mot de passe des données d'identification, les utilise pour la connexion et la validation auprès d'un serveur LDAP et crée un objet Subject résultant de l'authentification. L'implémentation peut utiliser les modules de connexion JAAS (Java Authentication and Authorization Service). Un objet Subject est retourné comme résultat de l'authentification.

Vous pouvez configurer l'authentificateur dans le fichier XML descripteur de sécurité, comme dans l'exemple qui suit :

```
<?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>
```

L'option **-clusterSecurityFile** lors du démarrage d'un serveur sécurisé permet de définir le fichier XML de sécurité. Pour plus d'informations, voir le tutoriel sur la sécurité Java SE dans la *Présentation du produit*.

Configuration d'un générateur de données d'identification système

Le générateur de données d'identification système permet de représenter la fabrique des données d'identification système. Les données d'identification système sont identiques aux données d'identification de l'administrateur. Vous pouvez configurer l'élément SystemCredentialGenerator dans le fichier XML de sécurité du catalogue :

```
<systemCredentialGenerator className ="com.ibm.websphere.objectgrid.security.plugins.
 builtins.UserPasswordCredentialGenerator">
          <property name="properties" type="java.lang.String" value="manager manager1"
      description="username password" />
 </systemCredentialGenerator>
```

Pour que cet exemple soit parlant, le nom d'utilisateur et le mot de passe sont indiqués en clair. Dans un environnement de production, ne stockez pas ces informations en clair.

WebSphere eXtreme Scale fournit un générateur de données d'identification système par défaut qui utilise les données d'identification du serveur. Si vous n'indiquez pas ce générateur de façon explicite, le générateur par défaut est utilisé.

Autorisation du client d'application

L'autorisation du client d'application consiste en des classes d'autorisation ObjectGrid, des mécanismes d'autorisation, une période de vérification des droits et une autorisation "accès réservé au créateur".

Pour eXtreme Scale, l'autorisation est accordée en fonction de l'objet et des droits du sujet. Le produit prend en charge deux sortes de mécanismes d'autorisation : le service JAAS (Java Authentication and Authorization Service) et l'autorisation personnalisée.

Classes d'autorisation ObjectGrid

L'autorisation est accordée en fonction des droits. Il existe quatre types de classes d'autorisation :

- La classe MapPermission représente les droits permettant d'accéder aux données des mappes ObjectGrid.
- La classe ObjectGridPermission représente les droits permettant d'accéder à ObjectGrid.
- La classe ServerMapPermission représente les droits permettant d'accéder aux mappes ObjectGrid se trouvant sur le serveur à partir du client.
- La classe AgentPermission représente les droits permettant de démarrer un agent se trouvant sur le serveur.

Pour plus d'informations sur les API et les droits associés, consultez la rubrique relative à la programmation des autorisations client dans le manuel *Guide de programmation*.

Période de vérification des droits

eXtreme Scale prend en charge la mise en cache des résultats de la vérification des droits d'accès aux mappes pour des raisons de performances. Sans ce mécanisme, l'environnement d'exécution appelle le mécanisme d'autorisation configuré pour autoriser l'accès lorsqu'une méthode répertoriée dans la liste des méthodes et des droits requis est appelée. Lorsque cette période est définie, le mécanisme d'autorisation est appelé périodiquement.

Les informations relatives à l'autorisation des droits se fondent sur l'objet Subject. Lorsqu'un client essaie d'accéder aux méthodes, l'environnement d'exécution eXtreme Scale recherche l'objet Subject dans le cache. Si cet objet est introuvable, l'environnement d'exécution vérifie les droits qui lui sont accordés, puis stocke les droits dans un cache.

La période de vérification des droits doit être définie avant l'initialisation de la grille d'objets. Vous pouvez la configurer de deux manières différentes :

Vous pouvez utiliser le fichier XML ObjectGrid pour définir une grille d'objets ainsi que la période de vérification des droits. Dans l'exemple suivant, cette période est définie pour une durée de 45 secondes :

```
<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 vous souhaitez créer une grille d'objets à l'aide des API, appelez la méthode suivante pour définir la période de vérification des droits. Cette méthode peut uniquement être appelée avant l'initialisation de l'instance ObjectGrid. Elle s'applique uniquement au modèle de programmation eXtreme Scale local lorsque vous instanciez directement une instance ObjectGrid.

```
/**
 * This method takes a single parameter indicating how often you
 * want to check the permission used to allow a client access. If the
 * parameter is 0 then every single get/put/update/remove/evict call
 * asks the authorization mechanism, either JAAS authorization or custom
 * authorization, to check if the current subject has permission. This might be
 * prohibitively expensive from a performance point of view depending on
 * the authorization implementation, but if you need to have ever call check the
 * authorization mechanism, then set the parameter to 0.
 * Alternatively, if the parameter is > 0 then it indicates the number
 * of seconds to cache a set of permissions before returning to
 * the authorization mechanism to refresh them. This value provides much
 * better performance, but if the back-end
 * permissions are changed during this time then the ObjectGrid can
 * allow or prevent access even though the back-end security
 * provider was modified.
 *
 * @param period the permission check period in seconds.
 */
void setPermissionCheckPeriod(int period);
```

Autorisation "accès réservé au créateur"

Avec l'autorisation "accès réservé au créateur", seul l'utilisateur (représenté par les objets Principal qui lui sont associés) ayant inséré une entrée dans une mappe ObjectGrid peut accéder (lecture, mise à jour, invalidation et suppression) à cette entrée.

Le modèle d'autorisation d'accès aux mappes ObjectGrid existant se fonde sur le type d'accès et non sur les entrées de données. En d'autres termes, un utilisateur dispose de droits d'accès d'un certain type (par exemple lecture, écriture, insertion, suppression ou invalidation) à toutes les données de la mappe ou ne détient aucun droit d'accès à aucune donnée. En revanche, eXtreme Scale n'autorise pas l'accès à certaines données seulement. Cette fonction constitue une nouvelle manière d'octroyer aux utilisateurs des droits d'accès aux entrées de données.

Dans un scénario où différents utilisateurs accèdent à différents jeux de données, ce modèle peut être utile. Lorsqu'un utilisateur charge des données à partir du stockage de persistence dans les mappes ObjectGrid, l'accès peut être autorisé par le stockage de persistence. Dans ce cas, il est inutile d'accorder une autre autorisation dans la couche de mappes ObjectGrid. Vous devez seulement faire en sorte que la personne qui charge les données dans la mappe peut y accéder en activant la fonction "réservé au créateur".

Valeurs des attributs en mode Réservé au créateur :

disabled
La fonction "accès réservé au créateur" est désactivée.

complement
La fonction "accès réservé au créateur" est activée et vient s'ajouter à l'autorisation d'accès aux mappes. En d'autres termes, les deux fonctions (autorisation d'accès aux mappes et fonction "accès réservé au créateur") sont opérationnelles. Vous pouvez donc limiter les opérations aux données. Le créateur ne peut par exemple pas invalider les données.

supersede
La fonction "accès réservé au créateur" est activée et remplace l'autorisation d'accès aux mappes. En d'autres termes, elle se substitue à cette autorisation, qui n'est plus opérationnelle.

Vous pouvez configurer le mode "accès réservé au créateur" de deux manières :

Avec un fichier XML :

Vous pouvez utiliser le fichier XML ObjectGrid pour définir une grille d'objets et choisir le mode `disabled`, `complement` ou `supersede`, comme dans l'exemple suivant :

```
<objectGrids>
    <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
         accessByCreatorOnlyMode="supersede"
        <bean id="TransactionCallback"
              classname="com.ibm.websphere.samples.objectgrid.HeapTransactionCallback" />
    ...
</objectGrids>
```

A l'aide d'un programme :

Si vous souhaitez créer une grille d'objets à l'aide d'un programme, vous pouvez appeler la méthode suivante pour définir le mode "accès réservé au créateur". L'appel de cette méthode s'applique uniquement au modèle de programmation eXtreme Scale local lorsque vous instanciez directement l'instance ObjectGrid :

```
/**
 * Set the "access by creator only" mode.
 * Enabling "access by creator only" mode ensures that only the user (represented
 * by the Principals associated with it), who inserts the record into the map,
 * can access (read, update, invalidate, and remove) the record.
 * The "access by creator only" mode can be disabled, or can complement the
 * ObjectGrid authorization model, or it can supersede the ObjectGrid
 * authorization model. The default value is disabled:
 * {@link SecurityConstants#ACCESS_BY_CREATOR_ONLY_DISABLED}.
```

```
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_DISABLED
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_COMPLEMENT
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_SUPERSEDE
 *
 * @param accessByCreatorOnlyMode the access by creator mode.
 *
 * @since WAS XD 6.1 FIX3
*/
void setAccessByCreatorOnlyMode(int accessByCreatorOnlyMode);
```

Autre exemple : imaginez un scénario selon lequel une grille de données bancaires contient un compte de mappe ObjectGrid dont les deux utilisateurs sont Manager1 et Employee1. Les règles d'autorisation d'eXtreme Scale accordent tous les droits d'accès à Manager1, mais uniquement les droits d'accès en lecture à Employee1. Les règles JAAS d'autorisation d'accès aux mappes ObjectGrid sont représentées ci-dessous :

```
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
    Principal com.acme.PrincipalImpl "Manager1" {
    permission com.ibm.websphere.objectgrid.security.MapPermission
        "banking.account", "all"
};
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
    Principal com.acme.PrincipalImpl "Employee1" {
    permission com.ibm.websphere.objectgrid.security.MapPermission
        "banking.account", "read, insert"
};
```

Observez quelle incidence la fonction "accès réservé au créateur" a sur l'autorisation :

- **disabled** Si la fonction "accès réservé au créateur" est désactivée, l'autorisation d'accès aux mappes reste inchangée. L'utilisateur "Manager1" peut accéder à toutes les données de la mappe "account". L'utilisateur "Employee1" peut lire toutes les données de la mappe et en insérer, mais il ne peut ni mettre à jour, ni invalider, ni supprimer ces données.
- **complement** Si la fonction "accès réservé au créateur" est activée avec l'option "complement", les deux fonctions sont opérationnelles. L'utilisateur "Manager1" peut accéder aux données de la mappe "account", mais uniquement s'il les a chargées dans celle-ci. L'utilisateur "Employee1" peut lire les données de cette mappe, mais uniquement s'il les a chargées. (Il ne peut cependant ni mettre à jour, ni invalider, ni supprimer les données de cette mappe.)
- **supersede** Si la fonction "accès réservé au créateur" est activée avec l'option "supersede", l'autorisation d'accès aux mappes n'est pas activée. L'autorisation "accès réservé au créateur" est alors la seule règle en vigueur. L'utilisateur "Manager1" détient les mêmes privilèges que ceux liés au mode "complement" : il peut accéder aux données de la mappe "account" uniquement s'il les a chargées dans la mappe. Toutefois, l'utilisateur "Employee1" détient maintenant les droits d'accès complet aux données de la mappe "account" s'il les a chargés dans la mappe. En d'autres termes, les règles d'autorisation définies dans les règles Java Authentication and Authorization Service (JAAS) ne sont pas appliquées.

Protocole TLS et couche de connexion sécurisée

WebSphere eXtreme Scale prend en charge les protocoles TCP/IP et TLS/SSL pour assurer une communication sécurisée entre les clients et les serveurs.

Le protocole TLS/SSL permet d'établir une communication sécurisée entre le client et le serveur. Le mécanisme de communication utilisé dépend de la valeur du paramètre transportType spécifiée dans les fichiers de configuration du client et du serveur.

Vous pouvez définir la propriété transportType dans les fichiers de configuration client et serveur suivants :

- Pour définir la propriété dans la configuration de sécurité du client, voir «Fichier de propriétés du client», à la page 205.
- Pour définir la propriété dans la configuration du serveur de conteneur, voir «Fichier de propriétés du serveur», à la page 185.
- Pour définir la propriété dans la configuration de sécurité du serveur de catalogue, voir «Fichier de propriétés du serveur», à la page 185.

Tableau 26. Protocole de transport à utiliser avec les paramètres de transport client et serveur

Propriété de client transportType	Propriété de serveur transportType	Résultat du protocole
TCP/IP	TCP/IP	TCP/IP
TCP/IP	SSL pris en charge	TCP/IP
TCP/IP	SSL requis	Erreur
SSL pris en charge	TCP/IP	TCP/IP
SSL pris en charge	SSL pris en charge	SSL (en cas d'échec du protocole SSL, TCP/IP)
SSL pris en charge	SSL requis	SSL
SSL requis	TCP/IP	Erreur
SSL requis	SSL pris en charge	SSL
SSL requis	SSL requis	SSL

Lorsque le protocole SSL est utilisé, les paramètres de configuration SSL doivent être fournis côté serveur et côté client. Dans un environnement Java SE, la configuration du protocole SSL s'effectue dans les fichiers de propriétés du client ou du serveur. Si le client ou le serveur se trouvent sur un serveur WebSphere Application Server, vous pouvez utiliser la prise en charge de la sécurité des transports de WebSphere Application Server pour configurer les paramètres SSL.

Configuration du fichier `orb.properties` pour la prise en charge de la sécurité des transports

Vous pouvez utiliser les protocoles TLS/SSL lorsque la propriété transportType est prise en charge par le protocole.

Pour assurer la prise en charge du transport sécurisé dans un environnement Java Platform, Standard Edition, vous devez modifier le fichier «Fichier de propriétés de l'ORB», à la page 195 pour qu'il inclue les propriétés suivantes :

```
# IBM JDK properties
org.omg.CORBA.ORBClass=com.ibm.CORBA.iiop.ORB
org.omg.CORBA.ORBSingletonClass=com.ibm.rmi.corba.ORBSingleton
javax.rmi.CORBA.StubClass=com.ibm.rmi.javax.rmi.CORBA.StubDelegateImpl
javax.rmi.CORBA.PortableRemoteObjectClass=com.ibm.rmi.javax.rmi.PortableRemoteObject
javax.rmi.CORBA.UtilClass=com.ibm.ws.orb.WSUtilDelegateImpl

# WS Plugins
com.ibm.CORBA.ORBPluginClass.com.ibm.ws.orbimpl.transport.WSTransport
com.ibm.CORBA.ORBPluginClass.com.ibm.ws.orbimpl.WSORBPropertyManager
com.ibm.CORBA.ORBPluginClass.com.ibm.ISecurityUtilityImpl.SecurityPropertyManager

# WS Interceptors
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.objectgrid.corba.ObjectGridInitializer
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ISecurityLocalObjectBaseL13Impl.SecurityComponentFactory

# WS ORB & Plugins properties
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
```

Configuration des paramètres SSL pour les clients eXtreme Scale

Vous pouvez configurer les paramètres SSL pour les clients à l'aide des méthodes suivantes :

1. Créez un objet com.ibm.websphere.objectgrid.security.config.SSLConfiguration à l'aide de la classe com.ibm.websphere.objectgrid.security.config. ClientSecurityConfigurationFactory. Pour plus d'informations, reportez-vous à la documentation d'API ClientSecurityConfigurationFactory.
2. Configurez les paramètres dans le fichier `client.properties`, puis utilisez la méthode ClientSecurityConfigurationFactory.getClientSecurityConfiguration(String) pour remplir l'instance d'objets.

Consultez la section sur les propriétés du client de sécurité dans «Fichier de propriétés du client», à la page 205 pour obtenir des exemples de propriétés que vous pouvez définir sur un client.

Configuration des paramètres SSL pour les serveurs eXtreme Scale

Les paramètres SSL sont configurés pour les serveurs à l'aide d'un fichier de propriétés de serveur, comme les exemples de fichiers `server.properties` ci-dessus. Ce fichier de propriétés peut être utilisé en tant que paramètre lors du démarrage d'un serveur eXtreme Scale. Pour obtenir des informations sur les paramètres SSL que vous pouvez définir pour les serveurs eXtreme Scale, voir «Fichier de propriétés du serveur», à la page 185.

Prise en charge de la sécurité des transports dans WebSphere Application Server

Lorsqu'un client eXtreme Scale, un serveur de conteneur ou un serveur de catalogue s'exécute dans un processus de serveur d'application WebSphere, la sécurité des transports eXtreme Scale est gérée par les paramètres de transport du serveur d'applications CSIV2. Pour le client ou le conteneur de serveurs eXtreme Scale, il est déconseillé d'utiliser les propriétés de client ou de serveur eXtreme Scale pour configurer les paramètres SSL. Tous les paramètres SSL doivent être spécifiés dans la configuration du serveur d'applications WebSphere.

Cependant, le serveur de catalogues est légèrement différent. Le serveur de catalogue dispose de ses propres chemins d'accès propriétaires, que les paramètres de transport du serveur d'application CSIV2 ne peuvent pas gérer. Par conséquent, il reste nécessaire de configurer les propriétés dans le fichier de propriétés du serveur pour le serveur de catalogue.

Activation de la sécurité de transport pour kit JDK Sun

WebSphere eXtreme Scale nécessite l'extension IBM Java Secure Sockets Extension (IBMJSSE) ou l'extension IBM Java Secure Sockets Extension 2 (IBMJSSE2). Les fournisseurs IBMJSSE et IBMJSSE2 contiennent une implémentation de référence prenant en charge les protocoles SSL et TLS, ainsi qu'un framework d'API.

Le kit JDK Sun pur n'inclut pas les fournisseurs JSSE IBM et JSSE2 IBM, par conséquent, la sécurité de transport ne peut pas être activée avec un kit JDK Sun. Pour que cela fonctionne, un kit JDK Sun fourni avec le serveur d'applications

WebSphere est requis. Le kit JDK Sun fourni avec le serveur d'applications WebSphere contient les fournisseurs IBM JSSE et IBM JSSE2.

Lisez la section sur la configuration d'une fonction ORB afin de pouvoir utiliser un kit JDK non IBM pour WebSphere eXtreme Scale. Si `-Djava.endorsed.dirs` est configuré, il pointe vers les répertoires `objectgridRoot/lib/endorsed` et `JRE/lib/endorsed`. Le répertoire `objectgridRoot/lib/endorsed` est requis de sorte que la fonction ORB IBM est utilisée et le répertoire `JRE/lib/endorsed` est requis pour le chargement des fournisseurs JSSE IBM et JSSE IBM.

Consultez l'étape 4 du tutoriel de sécurité du *Présentation du produit* pour obtenir des informations sur la configuration des propriétés SSL requises, la création de fichiers de clés et de clés certifiées, et le démarrage de serveurs sécurisés dans WebSphere eXtreme Scale.

Sécurité JMX (Java Management Extensions)

Vous pouvez sécuriser les invocations de beans gérés (MBean) dans un environnement réparti.

Pour plus d'informations sur les beans gérés disponibles, voir «Administration par programmation à l'aide de beans gérés (MBeans)», à la page 355.

Dans une topologie de déploiement réparti, les beans gérés sont directement hébergés sur les serveurs de catalogues et les serveurs conteneurs. En général, la sécurité JMX dans une topologie répartie suit les spécifications de sécurité JMX tel que spécifié dans les spécifications JMX (JavaTM Management Extensions). Elle est composée des trois parties suivantes :

1. Authentification : le client distant doit être authentifié dans le serveur de connecteur.
2. Contrôle d'accès : le contrôle de l'accès des beans gérés définit les privilèges d'accès aux informations de beans gérés et les droits d'exécution des opérations de beans gérés.
3. Transfert sécurisé : le transfert entre le client et le serveur JMX peut être sécurisé à l'aide du protocole TLS/SSL.

Authentification

JMX offre des méthodes aux serveurs de connecteur pour authentifier les clients distants. Pour le connecteur RMI, l'authentification est effectuée en fournissant un objet qui implémente l'interface JMXAuthenticator lors de la création du serveur de connecteur. Par conséquent,eXtreme Scale implémente cette interface JMXAuthenticator pour utiliser le plug-in de l'authentificateur ObjectGrid pour authentifier les clients distants. reportez-vous au didacticiel de sécurité de la *Présentation du produit* pour obtenir des détails sur le mode d'authentification d'un client par eXtreme Scale.

Le client JMX suit les API JMX pour offrir des données d'identification permettant la connexion au serveur de connecteur. L'infrastructure JMX transmet les données d'identification au serveur de connecteur et appelle l'implémentation JMXAuthenticator pour l'authentification. Tel que décrit précédemment, l'implémentation JMXAuthenticator délègue ensuite l'authentification à l'implémentation de l'authentificateur ObjectGrid.

Passez en revue l'exemple présenté ci-dessous, qui décrit comment établir la connexion à un serveur de connecteur à l'aide de données d'identification :

```
javax.management.remote.JMXServiceURL jmxUrl = new JMXServiceURL(
        "service:jmx:rmi:///jndi/rmi://localhost:1099/objectgrid/MBeanServer");

    environment.put(JMXConnector.CREDENTIALS, new UserPasswordCredential("admin", "xxxxxx"));

    // Créez le JMXCconnectorServer
    JMXConnector cntor = JMXConnectorFactory.newJMXConnector(jmxUrl, null);

    // Connectez et appelez une opération sur le MBeanServer distant
    cntor.connect(environment);
```

Dans l'exemple précédent, un objet UserPasswordCredential est fourni avec l'ID utilisateur défini sur et le mot de passe défini sur xxxxx. Cet objet UserPasswordCredential est défini dans la mappe d'environnement qui est utilisée dans la méthode JMXConnector.connect(Map). Cet objet UserPasswordCredential est ensuite transmis au serveur par l'infrastructure JMX, puis à l'infrastructure d'authentification ObjectGrid pour authentification.

Le modèle de programmation client respecte strictement les spécifications JMX.

Contrôle d'accès

Un serveur de beans gérés JMX peut avoir accès aux informations sensibles et peut être en mesure d'effectuer des opérations sensibles. JMX offre le contrôle d'accès requis permettant d'identifier les clients pouvant accéder à telles ou telles informations et qui peut effectuer ces opérations. Le contrôle d'accès repose sur le modèle de sécurité Java standard en définissant des autorisations de contrôle d'accès au serveur de beans gérés et aux opérations correspondantes.

Pour le contrôle d'accès ou l'autorisation des opérations JMX, eXtreme Scale repose sur le support JAAS fourni par l'implémentation JMX. A n'importe quel stade de l'exécution d'un programme, il existe un ensemble d'autorisations maintenu dans une unité d'exécution. Lorsqu'une unité d'exécution appelle une opération de spécification JMX, celle-ci est connue sous le terme d'autorisation de maintien. Lorsqu'une opération JMX est effectuée, une vérification de sécurité est réalisée pour vérifier si l'autorisation requise est concernée par l'autorisation de maintien.

La définition des règles d'administration de beans gérés respecte le format de la stratégie Java. Par exemple, la stratégie suivante octroie à tous les signataires et bases de code le droit d'extraire l'adresse JMX du serveur pour le PlacementServiceMBean, à l'exception du domaine com.ibm.websphere.objectgrid.

```
grant {
    permission javax.management.MBeanPermission
        "com.ibm.websphere.objectgrid.management.PlacementServiceMBean#retrieveServerJMXAddress
     [com.ibm.websphere.objectgrid:*,type=PlacementService]",
        "invoke";
}
```

Vous pouvez utiliser l'exemple de stratégie suivant pour compléter l'autorisation en fonction de l'identité du client distant. La stratégie octroie la même autorisation de bean géré que celle présentée dans l'exemple précédent, sauf pour les utilisateurs dont le nom X500Principal est 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";
}
```

Les stratégies Java sont uniquement vérifiées si le gestionnaire de sécurité est activé. Démarrez les serveurs de catalogues et les serveurs conteneurs à l'aide de l'argument JVM -Djava.security.manager pour forcer le contrôle d'accès des opérations de beans gérés.

Transfert sécurisé

Le transfert entre le client et le serveur JMX peut être sécurisé à l'aide du protocole TLS/SSL. Si le type de transfert du serveur de catalogues ou du serveur conteneur est défini sur SSL_Required ou SSL_Supported, vous devez utiliser le protocole SSL pour établir la connexion au serveur JMX.

Pour utiliser le protocole SSL, vous devez configurer le fichier de clés certifiées, le type du fichier de clés certifiées et le mot de passe du fichier de clés certifiées sur le client de bean géré à l'aide des propriétés -D system :

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 vous utilisez com.ibm.websphere.ssl.protocol.SSLSocketFactory comme fabrique de sockets SSL dans le fichier JAVA_HOME/jre/lib/security/java.security, utilisez les propriétés suivantes :

1. -Dcom.ibm.ssl.trustStore=TRUST_STORE_LOCATION
2. -Dcom.ibm.ssl.trustStorePassword=TRUST_STORE_PASSWORD
3. -Dcom.ibm.ssl.trustStoreType=TRUST_STORE_TYPE

Intégration de la sécurité à des fournisseurs externes

Pour protéger vos données WebSphere eXtreme Scale, eXtreme Scale peut intégrer plusieurs fournisseurs de sécurité.

WebSphere eXtreme Scale peut intégrer un programme de sécurité externe. Ce programme externe doit fournir des services d'authentification et d'autorisation pour eXtreme Scale. eXtreme Scale dispose de points de plug-in pour intégrer un programme de sécurité. WebSphere eXtreme Scale a été intégré aux composants suivants :

- Protocole LDAP (Lightweight Directory Access Protocol)
- Kerberos
- Sécurité ObjectGrid
- Tivoli Access Manager
- Service JAAS (Java Authentication and Authorization Service)

eXtreme Scale utilise le fournisseur de sécurité pour les tâches suivantes :

- Authentifier les clients sur les serveurs.
- Autoriser les clients à accéder à certains artefacts eXtreme Scale ou à préciser ce qui peut être fait avec les artefacts eXtreme Scale.

eXtreme Scale propose les types d'autorisations suivants :

Autorisation de mappes
Les clients et les groupes peuvent être autorisés à insérer, lire, mettre à jour, expulser ou supprimer des opérations sur les mappes.

Autorisation ObjectGrid
Les clients ou les groupes peuvent être autorisés à effectuer des requêtes de type objet ou entité sur les grilles d'objets.

Autorisation de l'agent DataGrid
Les clients ou les groupes peuvent être autorisés à permettre aux agents DataGrid d'être déployés en une base de données ObjectGrid.

Autorisation de mappes côté serveur
Les clients ou les groupes peuvent être autorisés à répliquer une mappe de serveur côté client ou à créer un index dynamique pour la mappe de serveur.

Autorisation d'administration
Les clients ou les groupes peuvent être autorisés à effectuer des tâches administratives.

Remarque : Si la sécurité est activée pour le dorsal, rappelez-vous que ces paramètres ne sont plus suffisants pour protéger vos données. Les paramètres de sécurité de votre base de données ou autre magasin de données ne sont transférés en aucune façon vers votre cache. Vous devez protéger séparément les données à présent mises en cache en utilisant le mécanisme de sécurité eXtreme Scale, qui inclut l'authentification, l'autorisation et la sécurité du niveau de transport.

Restriction : N'utilisez pas le JDK/jre 1.6 ou supérieur lorsque vous utilisez la sécurité SSL de la couche de transport avec un déploiement autonome de WeSbphere eXtreme Scale 7.1 (ou 7.0). Le JDK/jre 1.6 et versions supérieures ne prennent pas en charge les interfaces de programmes d'application WXS 7.1. Utilisez le JDK/jre 1.5 ou versions inférieures pour les configurations requérant la sécurité SSL de la couche de transport pour des installations autonomes d'eXtreme Scale. Cette recommandation ne s'applique qu'à l'utilisation de la sécurité SSL dans des configurations autonomes d'eXtreme Scale. Le JDK/jre n'est pas pris en charge pour les configurations non SSL de transport.

Intégration de la sécurité dans WebSphere Application Server

WebSphere eXtreme Scale fournit plusieurs fonctions de sécurité à intégrer dans l'infrastructure de sécurité de WebSphere Application Server.

Intégration de l'authentification

Lorsque les clients et serveurs eXtreme Scale sont exécutés dans WebSphere Application Server et dans le même domaine de sécurité, vous pouvez utiliser l'infrastructure de sécurité de WebSphere Application Server pour propager les données d'accès pour l'authentification du client sur le serveur eXtreme Scale. Par exemple, si un servlet agit en tant que client eXtreme Scale pour se connecter à un serveur eXtreme Scale du même domaine de sécurité et si le servlet est déjà authentifié, il est possible de propager le jeton d'authentification du client (servlet) vers le serveur, puis utiliser l'infrastructure de sécurité de WebSphere Application Server pour reconvertir ce jeton en données d'accès du client.

Intégration de la sécurité répartie dans WebSphere Application Server

Pour le modèle ObjectGrid réparti, l'intégration de la sécurité peut se faire à l'aide des classes suivantes :

```
com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator
com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator
```

```
com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredential
```

Pour plus d'informations, voir «Authentification du client d'application», à la page 360. L'exemple suivant indique comment utiliser la classe WSTokenCredentialGenerator :

```
/**
 * connect to the ObjectGrid Server.
 */
protected ClientClusterContext connect() throws ConnectException {
  ClientSecurityConfiguration csConfig = ClientSecurityConfigurationFactory
  .getClientSecurityConfiguration(proFile);

  CredentialGenerator gen = getWSCredGen();

  csConfig.setCredentialGenerator(gen);

  return objectGridManager.connect(csConfig, null);
}

/**
 * Get a WSTokenCredentialGenerator
 *
private CredentialGenerator getWSCredGen() {
    WSTokenCredentialGenerator gen = new WSTokenCredentialGenerator(
        WSTokenCredentialGenerator.RUN_AS_SUBJECT);
    return gen;
}
```

Côté serveur, utilisez l'authentificateur WSTokenAuthentication pour authentifier l'objet WSTokenCredential.

Intégration de la sécurité locale dans WebSphere Application Server

Pour le modèle ObjectGrid, l'intégration de la sécurité peut se faire à l'aide des deux classes suivantes :

- com.ibm.websphere.objectgrid.security.plugins.builtins.WSSubjectSourceImpl
- com.ibm.websphere.objectgrid.security.plugins.builtins.WSSubjectValidationImpl

Pour plus d'informations sur ces classes, voir les informations sur la sécurité locale dans le *Guide de programmation*. Vous pouvez configurer la classe WSSubjectSourceImpl en tant que plug-in SubjectSource et la classe WSSubjectValidationImpl en tant que plug-in SubjectValidation.

Démarrage et arrêt des serveurs sécurisés eXtreme Scale

Il est souvent nécessaire de sécuriser les serveurs pour votre environnement de déploiement. Pour cela, il faut configurer d'une certaine façon le démarrage et l'arrêt.

Démarrage d'un serveur de sécurité dans un environnement de système d'exploitation Java

Vous pouvez démarrer un service de catalogue ou des serveurs de conteneur comme décrit ci-dessous.

Démarrage d'un service de catalogue eXtreme Scale sécurisé

Le démarrage d'un processus de service de catalogue eXtreme Scale requiert deux fichiers de configuration de sécurité supplémentaires :

Fichier XML du descripteur de sécurité : le fichier XML du descripteur de sécurité décrit les propriétés de sécurité communes à tous les serveurs (y compris les

serveurs de catalogue et les serveurs de conteneur). Un exemple de propriété est la configuration de l'authentificateur qui représente le registre utilisateur et le mécanisme d'authentification.

Fichier de propriétés du serveur. Le fichier de propriétés du serveur configure les propriétés de sécurité spécifiques au serveur.

Lorsque vous utilisez la commande StartOgServer.sh ou startOgServer.cat pour démarrer un processus de service de catalogue eXtreme Scale sécurisé, vous pouvez utiliser -clusterSecurityFile ou -clusterSecurityUrl pour définir le fichier XML du descripteur de sécurité en tant que type de fichier ou d'URL. Vous pouvez également utiliser –serverProps pour paramétrer le fichier de propriétés du serveur.

Démarrage d'un serveur de conteneur eXtreme Scale sécurisé

Le démarrage d'un serveur de conteneur eXtreme Scale sécurisé requiert un fichier de configuration de sécurité :

- **Fichier de propriétés du serveur :** ce fichier permet de configurer les propriétés de sécurité spécifiques au serveur. Pour plus d'informations, voir «Fichier de propriétés du serveur», à la page 185.

Lorsque vous utilisez la commande startOgServer.sh ou startOgServer.cat pour démarrer le serveur de conteneur eXtreme Scale, vous pouvez utiliser –serverProps pour paramétrer le fichier de propriétés du serveur. Il existe plusieurs façons de paramétrer le fichier de propriétés du serveur. Référez-vous à ce fichier pour plus d'informations.

Pour plus d'informations sur la façon d'utiliser la commande startOgServer.sh ou startOgServer.bat et ses options, référez-vous à «Script startOgServer», à la page 336.

Arrêt d'un serveur eXtreme Scale sécurisé

L'arrêt d'un processus de service de catalogue eXtreme Scale sécurisé ou d'un serveur de conteneur requiert un fichier de configuration de sécurité :

- **Fichier de propriétés du client :** ce fichier permet configurer les propriétés de sécurité du client. Ces propriétés de sécurité permettent à un client de se connecter à un serveur sécurisé. Pour plus d'informations, référez-vous à «Fichier de propriétés du client», à la page 205.

Lorsque vous utilisez la commande stopOgServer.sh ou stopOgServer.cat pour arrêter un processus de service de catalogue eXtreme Scale ou un serveur de conteneur, vous pouvez utiliser -clientSecurityFile pour définir les propriétés de sécurité du client.

Pour plus d'informations sur la façon d'utiliser la commande stopOgServer.sh ou stopOgServer.cat et ses options, voir «Script stopOgServer», à la page 340.

Démarrage d'un serveur de sécurité dans WebSphere Application Server

Le démarrage d'un serveur ObjectGrid sécurisé dans WebSphere Application Server est similaire au démarrage d'un serveur ObjectGrids non sécurisé, à la différence près que vous devez passer les fichiers de configuration de sécurité. Au lieu d'utiliser l'argument -[FICHIER_PROPRIETES] (par exemple -serverProps) dans la

commande, comme c'est le cas dans un environnement Java SE, vous utilisez -D[FICHIER_PROPRIETES] dans les arguments génériques Java Virtual Machine (JVM).

Démarrage d'un service de catalogue sécurisé dans Websphere Application Server

Un serveur de catalogue contient deux niveaux d'informations de sécurité :

- `-Dobjectgrid.cluster.security.xml.url` : spécifie le fichier objectGridSecurity.xml décrivant les propriétés de sécurité communes à tous les serveurs (y compris les serveurs de catalogue et les serveurs de conteneur). Un exemple est la configuration de l'authentificateur, qui inclut le registre utilisateur et le mécanisme d'authentification. Le nom de fichier spécifié pour cette propriété doit être dans un format URL, tel que "file:///tmp/og/objectGridSecurity.xml".
- `-Dobjectgrid.server.props` : spécifie le fichier de propriétés du serveur, contenant les propriétés de sécurité spécifiques au serveur. Le nom de fichier spécifié pour cette propriété est au format de chemin de fichier simple, par exemple "c:/tmp/og/catalogserver.props". L'utilisation de -Dobjectgrid.security.server.props est dépréciée, mais vous pouvez continuer à l'utiliser pour la compatibilité rétroactive.

Pour démarrer un service de catalogue sécurisé dans WebSphere Application Server, suivez les instructions de la section "Intégré à WebSphere Application Server" dans «Sécurité de la grille», à la page 358.

Ensuite, définissez la propriété de sécurité dans l'argument JVM générique du processus.

```
-Dobjectgrid.cluster.security.xml.url=file:///tmp/og/
objectGridSecurity.xml-Dobjectgrid.server.props=/tmp/og/
catalog.server.props
```

Les étapes permettant d'ajouter les arguments JVM génériques sont les suivantes :

- Développez Administration système dans la vue des tâches gauche.
- Cliquez sur le processus WebSphere Application Server sur lequel est déployé le service de catalogue, par exemple "Gestionnaire de déploiement".
- Sur la page droite, développez "Gestion des processus et Java " sous "Infrastructure du serveur".
- Cliquez sur "Définition des processus".
- Cliquez sur "Java Virtual Machine" sous "Autres propriétés".
- Entrez les propriétés dans la zone de saisie Arguments JVM génériques.

Démarrage d'un serveur de conteneur sécurisé dans WebSphere Application Server

Lorsqu'il se connecte au serveur de catalogue, le serveur de conteneur obtient les paramètres de sécurité configurés dans objectGridSecurity.xml, tels que la configuration de l'authentificateur ou le délai d'attente de la session de connexion. De plus, les propriétés de sécurité spécifiques au serveur doivent être définies au niveau du serveur de conteneur, dans la propriété -Dobjectgrid.server.props.

Utilisez la propriété -Dobjectgrid.server.props property au lieu de la propriété -Dobjectgrid.security.server.propsproperty ; ce fichier de propriétés contient

également des propriétés non liées à la sécurité. Le nom de fichier spécifié pour cette propriété est au format de chemin de fichier simple, tel que `c:/tmp/og/server.props`.

Suivez la même procédure que ci-dessus pour ajouter les propriétés de sécurité aux arguments JVM génériques.

Fichier XML du descripteur de sécurité

Utilisez un fichier XML de descripteur de sécurité ObjectGrid pour configurer une topologie de déploiement eXtreme Scale avec la sécurité activée. Les exemples de fichiers XML suivants décrivent plusieurs configurations.

Chaque élément et attribut du fichier XML du cluster est décrit dans la liste suivante. Utilisez ces exemples pour apprendre à utiliser ces éléments et attributs et configurer l'environnement.

Elément securityConfig

L'élément securityConfig est l'élément de premier niveau du fichier XML de sécurité ObjectGrid. Cet élément définit l'espace de noms du fichier et l'emplacement du schéma. Le schéma est défini dans le fichier `objectGridSecurity.xsd`.

- nombre d'occurrences : une
- éléments enfants : security

Elément security

Utilisez l'élément security pour définir une sécurité ObjectGrid.

- nombre d'occurrences : une
- éléments enfants : authenticator, adminAuthorization et systemCredentialGenerator

Attributs

securityEnabled
Si la valeur est true, active la sécurité pour la grille. La valeur par défaut est false. Si la valeur est false, la sécurité est désactivée sur l'ensemble de la grille. Pour plus d'informations, voir «Sécurité de la grille», à la page 358 (facultatif).

singleSignOnEnabled
Si la valeur est true, permet à un client de se connecter à n'importe quel serveur après qu'il s'est authentifié sur l'un des serveurs. Dans le cas contraire, le client doit s'authentifier sur chaque serveur pour pouvoir se connecter. La valeur par défaut est false (facultatif).

loginSessionExpirationTime
Indique la durée, en secondes, avant que la session de connexion expire. Si la session de connexion expire, le client doit s'authentifier à nouveau (facultatif)

adminAuthorizationEnabled
Active l'autorisation administrative. Si la valeur est true, toutes les tâches administratives requièrent une autorisation. Le mécanisme d'autorisation utilisé est spécifié par la valeur de l'attribut adminAuthorizationMechanism. La valeur par défaut est false (facultatif).

adminAuthorizationMechanism
Indique le mécanisme d'autorisation à utiliser. WebSphere eXtreme Scale prend en charge deux mécanismes d'autorisation, le service JAAS (Java Authentication and Authorization Service) et l'autorisation personnalisée. Le mécanisme d'autorisation JAAS utilise l'approche basée sur la stratégie JAAS. Pour définir JAAS en tant que mécanisme d'autorisation, définissez la valeur sur AUTHORIZATION_MECHANISM_JAAS. Le mécanisme d'autorisation personnalisée fait appel à une implémentation AdminAuthorization connectée par l'utilisateur. Pour activer le mécanisme d'autorisation personnalisé, définissez la valeur sur AUTHORIZATION_MECHANISM_CUSTOM. Pour plus d'informations sur la façon dont ces deux mécanismes sont utilisés, voir «Autorisation du client d'application», à la page 362 (facultatif).

Le fichier `security.xml` suivant est un exemple de configuration permettant d'activer la sécurité de grille 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>
```

Elément authenticator

Authentifie les clients sur les serveurs eXtreme Scale dans la grille. La classe spécifiée par l'attribut className doit implémenter l'interface com.ibm.websphere.objectgrid.security.plugins.Authenticator. L'authentificateur peut utiliser les propriétés pour appeler des méthodes sur la classe spécifiée par l'attribut className. Reportez-vous à la section sur l'élément property pour en savoir plus sur l'utilisation des propriétés.

Dans l'exemple de fichier `security.xml` précédent, la classe com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator est définie en tant qu'authentificateur. Cette classe implémente l'interface com.ibm.websphere.objectgrid.security.plugins.Authenticator.

- nombre d'occurrences : zéro ou une
- élément enfant : property

Attributs

className
Spécifie une classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.Authenticator. Utilisez cette classe pour authentifier les clients sur les serveurs dans la grille eXtreme Scale. (Obligatoire)

Elément adminAuthorization

Utilisez l'élément adminAuthorization pour configurer un accès administratif à la grille.

- nombre d'occurrences : zéro ou une
- élément enfant : property

Attributs

className
Spécifie une classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.AdminAuthorization. (Obligatoire)

Elément systemCredentialGenerator

Utilisez un élément systemCredentialGenerator pour configurer un générateur de données d'identification système. Cet élément s'applique uniquement à un environnement dynamique. Dans le modèle de configuration dynamique, le serveur de conteneur dynamique se connecte au serveur de catalogue en tant que client eXtreme Scale et le serveur de catalogue peut se connecter au serveur de conteneur eXtreme Scale en tant que client également. Le générateur de données d'identification système représente une fabrique pour les données d'identification système.

- nombre d'occurrences : zéro ou une
- élément enfant : property

Attributs

className
Spécifie une classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. (Obligatoire)

Reportez-vous au précédent exemple de fichier `security.xml` pour savoir comment utiliser un systemCredentialGenerator. Dans cet exemple, le générateur de données d'identification système est com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator, qui récupère l'objet RunAs Subject à partir de l'unité d'exécution.

Elément property

Appelle les méthodes set sur les classes authenticator et adminAuthorization. Le nom de la propriété correspond à une méthode set sur l'attribut className de l'élément authenticator ou adminAuthorization.

- nombre d'occurrences : zéro ou plus
- élément enfant : property

Attributs

name
Indique le nom de la propriété. La valeur affectée à cet attribut doit correspondre à une méthode set sur la classe fournie en tant qu'attribut className du bean. Par exemple, si l'attribut className du bean est défini sur com.ibm.MyPlugin, et si le nom de la propriété fournie est size, alors la classe com.ibm.MyPlugin doit avoir une méthode setSize. (Obligatoire)

type
Indique le type de la propriété. Le type du paramètre est transmis à la méthode set identifiée par l'attribut name. Les valeurs valides sont les primitives Java et leur équivalent java.lang, java.lang.String. Les attributs name et type doivent correspondre à une signature de méthode sur l'attribut className du bean. Par exemple, si le nom est size et le type est int, alors une méthode setSize(int) doit exister sur la classe définie en tant qu'attribut className pour le bean. (Obligatoire)

value
Indique la valeur de propriété. Cette valeur est convertie vers le type défini par l'attribut type, puis est utilisée en tant que paramètre de l'appel de la méthode set identifiée par les attributs name et type. La valeur de cet attribut n'est validée en aucune façon. L'implémenteur du plug-in doit vérifier que la valeur transmise est valide. (Obligatoire)

description
Fournit une description de la propriété (facultatif).

Pour plus d'informations, voir «Fichier `objectGridSecurity.xsd`».

Fichier `objectGridSecurity.xsd`

Utilisez le schéma XML de sécurité ObjectGrid ci-après pour activer la sécurité sur un déploiement eXtreme Scale.

Pour les descriptions des éléments et des attributs définis dans le fichier `objectGridSecurity.xsd`, reportez-vous à la rubrique «Fichier XML du descripteur de sécurité», à la page 375.

```
<?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>
```

Chapitre 9. Surveillance de l'environnement de déploiement

Vous pouvez utiliser des API, des beans gérés, des journaux et des utilitaires pour surveiller les performances de votre environnement d'application.

Présentation des statistiques

Les statistiques dans WebSphere eXtreme Scale sont basés sur une arborescence interne de statistiques. L'API StatsAccessor, les modules PMI (Performance Monitoring Infrastructure) et l'API MBean sont générés à partir de l'arborescence interne.

La figure suivante affiche la configuration générale des statistiques pour eXtreme Scale.

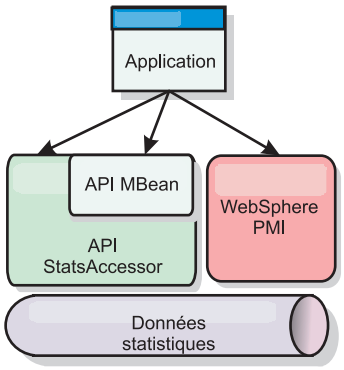


Figure 26. Présentation des statistiques

Toutes ces API permettent de visualiser l'arborescence des statistiques, mais chacune d'entre elles possède une fonction spécifique :

- **API Statistics :** cette API permet aux développeurs d'accéder directement aux statistiques, ce qui leur permet de réaliser des solutions flexibles et personnalisables d'intégration de statistiques, comme des beans gérés personnalisables ou de la journalisation.
- **API MBean :** cette API est un mécanisme de surveillance basé sur une spécification. Elle utilise l'API Statistics et s'exécute de manière locale sur la machine virtuelle Java du serveur. Les structures de l'API et des beans gérés sont conçues pour s'intégrer aisément à des utilitaires tiers. Utilisez l'API MBean lorsque vous exécutez une grille d'objets répartie.
- **WebSphere Application Server Modules PMI :** utilisez ces modules si vous exécutez WebSphere eXtreme Scale dans WebSphere Application Server. Ces modules permettent de visualiser l'arborescence interne des statistiques.

API Statistics

A l'instar d'une mappe d'arborescence, il existe un chemin et une clé correspondants qui permettent d'extraire un module spécifique ou, dans ce cas, le niveau de granularité ou d'agrégation. Par exemple, supposons que l'arborescence contienne toujours un nœud racine arbitraire et que les statistiques soient regroupées pour une mappe appelée "payroll" appartenant à une instance

d'ObjectGrid appelée "accounting". Par exemple, pour accéder au module en fonction du niveau d'agrégation ou de granularité d'une mappe, vous pouvez insérer un paramètre String[] des chemins. Dans ce cas, vous obtenez String[] {root, "accounting", "payroll"}, chaque paramètre String représentant le chemin du nœud. Cette structure a pour avantage de permettre à l'utilisateur de spécifier le tableau dans un nœud quelconque du chemin et d'obtenir le niveau d'agrégation du nœud en question. L'insertion du paramètre String[] {root, "accounting"} vous permet d'obtenir les statistiques de mappe, sauf pour la grille entière de "accounting". L'utilisateur peut ainsi spécifier les types de statistiques à surveiller, ainsi que le niveau d'agrégation nécessaire pour l'application.

WebSphere Application ServerModules PMI

WebSphere eXtreme Scale inclut des modules de statistiques à utiliser avec l'infrastructure PMI WebSphere Application Server. Lorsqu'une instance de WebSphere eXtreme Scale est ajoutée à un profil WebSphere Application Server, les scripts d'ajout intègrent automatiquement les modules WebSphere eXtreme Scale dans les fichiers de configuration WebSphere Application Server. PMI vous permet d'activer et de désactiver les modules de statistiques, d'assembler automatiquement les statistiques selon différents niveaux de granularité et même de représenter les données sous forme de graphiques à l'aide du logiciel pré-intégré Tivoli Performance Viewer. Pour plus d'informations, voir «Surveillance à l'aide de la fonction PMI de WebSphere Application Server», à la page 390.

Intégration de produits tiers avec les beans gérés (MBean)

Les API eXtreme Scale et les beans gérés sont conçus pour faciliter l'intégration d'applications de surveillance tierces. JConsole et MC4J sont des exemples de consoles Java Management Extensions (JMX) légères qui permettent d'analyser les informations relatives à une topologie eXtreme Scale. Vous avez également la possibilité d'utiliser les API de programmation pour écrire des implémentations d'adaptateur afin de créer des instantanés ou d'effectuer un suivi des performances d'eXtreme Scale. WebSphere eXtreme Scale inclut un exemple d'application de surveillance qui permet d'effectuer la surveillance dès l'installation et qui peut servir de modèle pour créer des utilitaires de surveillance plus avancés.

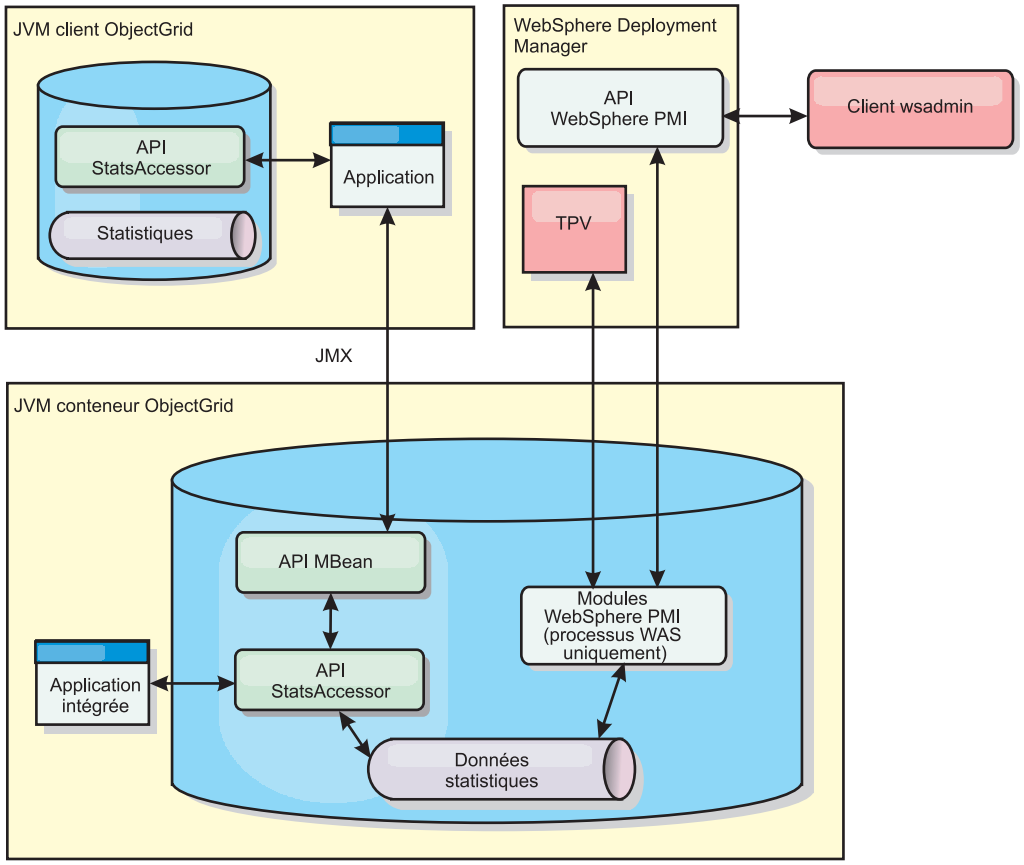


Figure 27. Présentation de l'API de bean géré

Pour plus d'informations, voir «Surveillance à l'aide de l'exemple d'utilitaire xsAdmin», à la page 387. Pour plus d'informations sur l'intégration d'applications tierces spécifiques, voir les rubriques suivantes :

- Surveillance d'eXtreme Scale à l'aide d'un agent de surveillance IBM Tivoli
- «Surveillance d'eXtreme Scale à l'aide de Hyperic HQ», à la page 420
- «Surveillance des applications eXtreme Scale à l'aide de CA Wily Introscope», à la page 416

Surveillance à l'aide de l'API Statistics

L'API Statistics est l'interface directe avec l'arborescence interne des statistiques. Les statistiques sont désactivées par défaut, mais peuvent être activées en définissant une interface StatsSpec. Une interface StatsSpec définit la manière dont WebSphere eXtreme Scale doit surveiller les statistiques.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser l'API locale StatsAccessor pour interroger les données et accéder aux statistiques d'une instance ObjectGrid qui se trouve sur la même machine virtuelle Java (JVM) que le code en cours d'exécution. Pour plus d'informations sur les interfaces spécifiques, voir la documentation de l'API. Utilisez les étapes ci-après pour activer la surveillance de l'arborescence des statistiques interne.

Procédure

1. Extrayez l'objet StatsAccessor. L'interface StatsAccessor suit le modèle des singletons. Par conséquent, en dehors des problèmes liés au chargeur de classe, il doit exister une instance StatsAccessor pour chaque JVM. Cette classe sert d'interface principale pour toutes les opérations sur les statistiques locales. Le code ci-après illustre l'extraction de la classe de l'accesseur. Appelez cette opération avant tout autre appel ObjectGrid.

```
public class LocalClient
{

   public static void main(String[] args) {

      // extrayez un descripteur de StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

   }

}
```

2. Définissez l'interface StatsSpec de la grille. Définissez cette JVM de sorte qu'elle ne collecte toutes les statistiques qu'au niveau d'ObjectGrid. Vous devez vérifier qu'une application active toutes les statistiques qui peuvent être requises avant de commencer des transactions. L'exemple ci-après définit l'interface StatsSpec à l'aide d'un champ de constante statique et d'une chaîne de spécification. L'utilisation d'un champ de constante statique est plus simple car le champ a déjà défini la spécification. Toutefois, en utilisant une chaîne de spécification, vous pouvez autoriser toutes les combinaisons de statistiques requises.

```
public static void main(String[] args) {

      // extrayez un descripteur de StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Définissez la spécification via le champ statique
      StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);
      accessor.setStatsSpec(spec);

      // Définissez la spécification via la chaîne de spécification
      StatsSpec spec = new StatsSpec("og.all=enabled");
      accessor.setStatsSpec(spec);

}
```

3. Envoyez des transactions à la grille pour force la collecte des données en vue de la surveillance. Pour collecter des données utiles pour les statistiques, vous devez envoyer des transactions à la grille. L'extrait de code suivant insère un enregistrement dans MapA, qui se trouve dans ObjectGridA. Les statistiques se trouvant au niveau d'ObjectGrid, toute mappe dans l'ObjectGrid renvoie les mêmes résultats.

```
public static void main(String[] args) {

      // extrayez un descripteur de StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Définissez la spécification via le champ
      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");
```

```
      // Effectuez une insertion
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();
}
```

4. Interrogez un objet StatsFact à l'aide de l'API StatsAccessor. Tous les chemins d'accès aux statistiques sont associés à une interface StatsFact. L'interface StatsFact est une marque de réservation générique permettant d'organiser et d'inclure un objet StatsModule. Pour que vous puissiez accéder au véritable module de statistiques, l'objet StatsFact doit être extrait.

```
public static void main(String[] args)
{

      // extrayez un descripteur de StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Définissez la spécification via le champ statique
      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");

      // Effectuez une insertion
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();

      // Extrayez StatsFact

      StatsFact fact = accessor.getStatsFact(new String[] {"EmployeeGrid"},
    StatsModule.MODULE_TYPE_OBJECT_GRID);

}
```

5. Interagissez avec l'objet StatsModule. L'objet StatsModule est contenu dans l'interface StatsFact. Vous pouvez obtenir une référence au module à l'aide de l'interface StatsFact. L'interface StatsFact étant une interface générique, vous devez transtyper le module renvoyé dans le type StatsModule attendu. Cette tâche collectant des statistiques eXtreme Scale, l'objet StatsModule renvoyé est transtypé dans un type OGStatsModule. Une fois que le module est transtypé, vous avez accès à toutes les statistiques disponibles.

```
public static void main(String[] args) {

      // extrayez un descripteur de StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Définissez la spécification via le champ statique
      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");

      // Effectuez une insertion
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();
```

```
        // Extrayez StatsFact
        StatsFact fact = accessor.getStatsFact(new String[] {"EmployeeGrid"},
      StatsModule.MODULE_TYPE_OBJECT_GRID);

        // Extrayez le module et l'heure
        OGStatsModule module = (OGStatsModule)fact.getStatsModule();
        ActiveTimeStatistic timeStat =
      module.getTransactionTime("Default", true);
        double time = timeStat.getMeanTime();

  }
```

Modules des statistiques

WebSphere eXtreme Scale utilise un modèle de statistiques interne pour suivre et filtrer les données. Toutes les vues de données se basent sur cette structure sous-jacente pour assembler des instantanés des statistiques. Vous pouvez extraire des informations des modules de statistiques à l'aide de plusieurs méthodes.

Présentation

Dans WebSphere eXtreme Scale, les statistiques sont suivies et stockées dans des composants StatsModules. Le modèle de statistiques contient plusieurs types de modules :

OGStatsModule
: Fournit des statistiques sur une instance ObjectGrid, notamment le temps de réponse des transactions.

MapStatsModule
: Fournit des statistiques sur une mappe unique, notamment le nombre d'entrées et le taux de réussite.

QueryStatsModule
: Fournit des statistiques sur les requêtes, notamment la création de plan et les temps d'exécution.

AgentStatsModule
: Fournit des statistiques sur les agents d'API DataGrid, notamment les temps de sérialisation et d'exécution.

HashIndexStatsModule
: Fournit des statistiques sur la requête HashIndex et les temps d'exécution de maintenance.

SessionStatsModule
: Fournit des statistiques sur le plug-in du gestionnaire de sessions HTTP.

Pour de plus amples informations sur les modules de statistiques, voir le module com.ibm.websphere.objectgrid.stats dans la documentation d'API.

Statistiques dans un environnement local

Le modèle est structuré comme un arbre n-aire (une arborescence dont tous les nœuds sont au même degré) contenant tous les types de modules de statistiques répertoriés dans la liste précédente. Du fait de cette structure, tous les nœuds de l'arborescence sont représentés par l'interface StatsFact. L'interface StatsFact peut représenter un seul module ou un groupe de modules à des fins d'agrégation. Par exemple, si plusieurs nœuds terminaux de l'arborescence représentent des objets MapStatsModule spécifiques, le nœud StatsFact parent de ces nœuds contient les

statistiques agrégés pour tous les modules enfants. Une fois l'objet StatsFact extrait, vous pouvez extraire le module de statistiques correspondant à l'aide de l'interface.

A l'instar d'une mappe d'arborescence, vous pouvez utiliser un chemin ou une clé correspondante pour extraire un objet StatsFact spécifique. Le chemin est une valeur String[] qui contient tous les nœuds du chemin de l'objet demandé. Par exemple, supposons que vous avez créé un objet ObjectGrid appelé ObjectGridA, qui contient deux mappes : MapA et MapB. Le chemin du module de statistiques de MapA se présente comme suit : `[ObjectGridA, MapA]`. Le chemin des statistiques agrégées des deux mappes se présente comme suit : `[ObjectGridA]`.

Statistiques dans un environnement réparti

Dans un environnement réparti, les modules de statistiques sont extraits à l'aide d'un chemin différent. Un serveur pouvant contenir plusieurs partitions, l'arborescence de statistiques doit suivre la partition à laquelle chaque module appartient. Le chemin de recherche d'un objet StatsFact spécifique est donc différent. A l'aide de l'exemple précédent, en indiquant que les mappes se trouvent dans la partition 1, utilisez le chemin `[1, ObjectGridA, MapA]` afin d'extraire cet objet StatsFact pour MapA.

Surveillance à l'aide de l'exemple d'utilitaire xsAdmin

L'exemple d'utilitaire xsAdmin vous permet de mettre en forme et d'afficher les informations textuelles relatives à votre topologie WebSphere eXtreme Scale. Il fournit une méthode pour effectuer l'analyse syntaxique et la détection des données de déploiement actuelles et peut servir de base pour l'écriture d'utilitaires personnalisés.

Avant de commencer

WebSphere eXtreme Scale doit être installé.

Pourquoi et quand exécuter cette tâche

Vous pouvez utiliser l'exemple d'utilitaire xsAdmin pour apporter des retours d'informations sur la disposition actuelle et l'état spécifique de la grille, comme le contenu de la mappe. Dans cet exemple, la disposition de la grille de cette tâche se compose d'une seule grille, nommée *ObjectGridA* et d'une mappe définie, nommée *MapA* et appartenant au groupe de mappes, intitulé *MapSetA*. Cet exemple montre comment afficher tous les conteneurs actifs dans une grille et imprimer les métriques filtrées en fonction de la taille de *MapA*. Pour voir toutes les options de commande possibles, exécutez l'utilitaire xsAdmin sans argument ou avec l'option **-help**.

Procédure

1. Sur la ligne de commande, définissez la variable d'environnement JAVA_HOME.
   - UNIX `export JAVA_HOME=javaHome`
   - Windows `set JAVA_HOME=javaHome`
2. Accédez au répertoire bin.
   ```
   cd objectGridRoot/bin
   ```
3. Lancez l'utilitaire xsAdmin.
   - **Pour afficher l'aide en ligne, exécutez la commande suivante :**

UNIX

```
xsadmin.sh
```

Windows

```
xsadmin.bat
```

Notez bien la section des arguments obligatoires du message d'aide, car vous devez sélectionner une seule des options répertoriées pour que l'utilitaire fonctionne. Si aucune option **-g** ou **-m** n'est spécifiée, l'utilitaire xsAdmin imprime les informations pour chaque grille de la topologie.

- **POur activer les statistiques pour l'ensemble des serveurs, exécutez la commande suivante :**

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -setstatsspec ALL=enabled
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -setstatsspec ALL=enabled
  ```

- **Pour afficher tous les conteneurs en ligne pour une grille, exécutez la commande suivante :**

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -m MapSetA -containers
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -m MapSetA -containers
  ```

  Toutes les informations sur les conteneurs s'affichent. Ci-après, un exemple de sortie :

  ```
  Cet utilitaire administratif est donné uniquement en exemple et ne doit pas
  être considéré comme un composant pris en charge par le
  produit WebSphere eXtreme Scale

  Connecting to Catalog service at localhost:1099


  *** Show all online containers for grid - ObjectGridA & mapset - MapSetA

  Host: 192.168.0.186
  Container: server1_C-0, Server:server1, Zone:DefaultZone
  Partition Shard Type
          0 Primary

  Num containers matching = 1
  Total known containers = 1
  Total known hosts = 1
  ```

- **Pour vous connecter au service de catalogue et afficher les informations concernant MapA, exécutez la commande suivante :**

  UNIX

  ```
  xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  ```

  Windows

  ```
  xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  ```

  La taille de la mappe spécifiée s'affiche. Ci-après, un exemple de sortie :

  ```
  Cet utilitaire administratif est donné uniquement en exemple et ne doit pas
  être considéré comme un composant pris en charge par le
  produit WebSphere eXtreme Scale
  ```

```
Connecting to Catalog service at localhost:1099

****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


*** Listing Maps for server1 ***
 Map Name Partition Map Size Used Bytes (B) Shard Type
 MapA     0         0        0              Primary
```

- **Pour vous connecter au service de catalogue à l'aide d'un port JMX spécifique et afficher les informations concernant MapA, exécutez la commande suivante :** 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
```

L'utilitaire xsAdmin se connecte au serveur des beans gérés qui s'exécute dans un serveur de catalogue. Un serveur de catalogue peut être exécuté sur un processus autonome, processus WebSphere Application Server, ou imbriqué dans un processus applicatif personnalisé. Utilisez l'option **-ch** pour spécifier le nom d'hôte du service de catalogue et l'option **-p** pour spécifier son port de désignation.

La taille de la mappe spécifiée s'affiche. Ci-après, un exemple de sortie :

```
Cet utilitaire administratif est donné uniquement en exemple et ne doit pas
être considéré comme un composant pris en charge par le
produit WebSphere eXtreme Scale

Connecting to Catalog service at CatalogMachine:6645

*****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


*** Listing Maps for server1 ***
Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
Server Total: 0
```

- **Pour vous connecter à un service de catalogue hébergé sur un processus WebSphere Application Server, effectuez la procédure suivante :**

  L'option **-dmgr** est obligatoire lorsque vous vous connectez à un service de catalogue hébergé par tout processus ou cluster de processus WebSphere Application Server. Utilisez l'option **-ch** pour spécifier le nom d'hôte, s'il n'est pas `localhost`, et l'option **-p** pour substituer le port d'amorce du service de catalogue, qui utilise le processus BOOTSTRAP_ADDRESS. L'option **-p** est nécessaire uniquement si ce processus n'est pas défini sur la valeur par défaut 9809.

  Remarque : La version autonome de WebSphere eXtreme Scale ne peut pas se connecter à un service de catalogue hébergé par un processus WebSphere Application Server. Utilisez le script xsAdmin inclus dans le répertoire *was_root*`/bin`, qui est disponible lors de l'installation de WebSphere eXtreme Scale sur WebSphere Application Server ou WebSphere Application Server Network Deployment.

  a. Accédez au répertoire bin de WebSphere Application Server :

     ```
     cd wasRoot/bin
     ```

  b. Lancez xsAdmin à l'aide de la commande suivante :

UNIX

```
xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
```

Windows

```
xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
```

La taille de la mappe spécifiée s'affiche.

```
Cet utilitaire administratif est donné uniquement en exemple et ne doit pas
être considéré comme un composant pris en charge par le
produit WebSphere eXtreme Scale

Connecting to Catalog service at localhost:9809

****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


*** Listing Maps for server1 ***
Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
Server Total: 0
```

Surveillance à l'aide de la fonction PMI de WebSphere Application Server

WebSphere eXtreme Scale prend en charge PMI (Performance Monitoring Infrastructure) lorsqu'il est exécuté dans un serveur d'applications WebSphere Application Server ou WebSphere Extended Deployment. PMI collecte des données de performances relatives aux applications exécutables et offre des interfaces permettant aux applications externes de surveiller les données de performances. Vous pouvez utiliser la console d'administration ou l'outil wsadmin pour accéder aux données de surveillance.

Avant de commencer

Vous pouvez utiliser PMI pour surveiller votre environnement lorsque vous utilisez WebSphere eXtreme Scale avec WebSphere Application Server.

Pourquoi et quand exécuter cette tâche

WebSphere eXtreme Scale utilise la fonction PMI personnalisée de WebSphere Application Server pour ajouter sa propre instrumentation de PMI. Avec cette approche, vous pouvez activer et désactiver la fonction PMI de WebSphere eXtreme Scale à l'aide de la console d'administration ou des interfaces JMX (Java Management Extensions) de l'outil wsadmin. En outre, vous pouvez accéder aux statistiques de WebSphere eXtreme Scale à l'aide des interfaces PMI et JMX standard utilisées par les outils de surveillance et notamment Tivoli Performance Viewer.

Procédure

1. Activez la fonction PMI de eXtreme Scale. Vous devez activer PMI pour afficher les statistiques PMI. Pour plus d'informations, voir «Activation de PMI», à la page 391.
2. Extrayez les statistiques PMI de eXtreme Scale. Affichez les performances de vos applications eXtreme Scale à l'aide de Tivoli Performance Viewer. Pour plus d'informations, voir «Récupération des statistiques PMI», à la page 393.

Que faire ensuite

Pour plus d'informations sur l'outil wsadmin, voir «Accès aux beans gérés à l'aide de l'outil wsadmin», à la page 356.

Activation de PMI

Vous pouvez utiliser l'infrastructure PMI (Performance Monitoring Infrastructure) de WebSphere Application Server pour activer ou désactiver les statistiques à tout niveau. Par exemple, vous pouvez choisir d'activer les statistiques du nombre d'occurrences d'une mappe donnée, mais non le nombre de statistiques en entrée ou les statistiques de durée de mise à jour par lots du chargeur. Vous pouvez activer PMI dans la console d'administration ou à l'aide de scripts.

Avant de commencer

Votre serveur d'applications doit être démarré et une application compatible eXtreme Scale doit y être installée. Pour activer PMI à l'aide de scripts, vous devez pouvoir vous connecter et utiliser l'outil wsadmin. Pour plus d'informations sur l'outil wsadmin, reportez-vous à la rubrique Outil wsadmin, dans le Centre de documentation de WebSphere Application Server.

Pourquoi et quand exécuter cette tâche

Utilisez l'infrastructure PMI de WebSphere Application Server pour fournir un mécanisme granulaire à l'aide duquel vous pouvez activer ou désactiver les statistiques à tout niveau. Par exemple, vous pouvez choisir d'activer les statistiques du nombre d'occurrences d'une mappe donnée, mais non le nombre d'entrées ou les statistiques de durée de mise à jour par lots du chargeur. Cette section montre comment utiliser la console d'administration et les scripts wsadmin pour activer l'infrastructure PMI d'ObjectGrid.

Procédure

- **Activez PMI dans la console d'administration.**
  1. Dans la console d'administration, cliquez sur **Contrôle et réglage** → **Performance Monitoring Infrastructure** → *nom_serveur*.
  2. Vérifiez que la case Activer l'infrastructure PMI (Performance Monitoring Infrastructure) est cochée. Ce paramètre est activé par défaut. S'il ne l'est pas, cochez la case et redémarrez le serveur.
  3. Cliquez sur **Personnalisé**. Dans l'arborescence de configuration, sélectionnez l'ObjectGrid et le module Mappes d'ObjectGrid. Activez les statistiques de chaque module.

  La catégorie des types de transaction des statistiques ObjectGrid est créée lors de la phase d'exécution. Vous ne pouvez voir que les sous-catégories des statistiques ObjectGrid et des statistiques de mappe dans la page **Exécution**.
- **Activez PMI à l'aide de scripts.**
  1. Ouvrez une invite de ligne de commande. Accédez au répertoire racine_install/bin. Entrez wsadmin pour démarrer l'outil de ligne de commande wsadmin.
  2. Modifiez la configuration de l'environnement d'exécution de l'infrastructure PMI d'eXtreme Scale. Vérifiez que PMI est activé pour le serveur à l'aide des commandes suivantes :

```
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 n'est pas activé, exécutez les commandes suivantes pour activer PMI :

```
wsadmin>$AdminConfig modify $pmi {{enable true}}
wsadmin>$AdminConfig save
```

Si vous avez besoin d'activer PMI, redémarrez le serveur.

3. Définissez des variables pour modifier l'ensemble de statistiques en ensemble personnalisé à l'aide des commandes suivantes :

```
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. Spécifiez un ensemble de statistiques personnalisé à l'aide de la commande suivante :

```
wsadmin>$AdminControl invoke_jmx $perfOName setStatisticSet $params $sigs
```

5. Définissez des variables pour activer les statistiques de l'infrastructure PMI d'objectGridModule à l'aide des commandes suivantes :

```
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. Définissez la chaîne des statistiques à l'aide de la commande suivante :

```
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. Définissez la chaîne des statistiques à l'aide de la commande suivante :

```
wsadmin>$AdminControl invoke_jmx $perfOName setCustomSetString $params2 $sigs2
```

Ces étapes activent l'infrastructure PMI de l'environnement d'exécution d'eXtreme Scale, mais ne modifient pas la configuration de l'infrastructure PMI. Si vous redémarrez l'application, les paramètres PMI sont perdus, exceptée l'activation principale de PMI.

Exemple

Vous pouvez effectuer les étapes suivantes pour activer les statistiques PMI de l'exemple d'application :

1. Lancez l'application à l'aide de l'adresse Web `http://hôte:port/ObjectGridSample`, hôte et port correspondant au nom d'hôte et au numéro de port HTTP du serveur où l'exemple est installé.
2. Dans l'exemple d'application, cliquez sur ObjectGridCreationServlet, puis sur les boutons d'action 1, 2, 3, 4 et 5 pour générer des actions sur l'ObjectGrid et les mappes. Ne fermez pas tout de suite cette page de servlet.

3. Dans la console d'administration, cliquez sur **Contrôle et réglage** → **Performance Monitoring Infrastructure** → *nom_serveur*. Cliquez sur l'onglet **Exécution**.
4. Cliquez sur le bouton d'option **Personnalisé**.
5. Développez le module Mappes d'ObjectGrid dans l'arborescence d'exécution, puis cliquez sur le lien clusterObjectGrid. Le groupe Mappes d'ObjectGrid contient une instance ObjectGrid appelée clusterObjectGrid et le groupe clusterObjectGrid contient quatre mappes : counters, employees, offices, et sites. Dans l'instance ObjectGrids se trouve une instance clusterObjectGrid et sous cette instance, le type de transaction DEFAULT.
6. Vous pouvez activer les statistiques de votre choix. Par exemple, vous pouvez activer le nombre d'entrées de mappe pour la mappe des employés et le temps de réponse des transactions pour le type de transaction DEFAULT.

Que faire ensuite

Une fois que PMI est activé, vous pouvez afficher les statistiques PMI à l'aide de la console d'administration ou de scripts.

Récupération des statistiques PMI

En récupérant les statistiques PMI, vous pouvez voir les performances de vos applications eXtreme Scale.

Avant de commencer

- Activez la fonction de suivi des statistiques PMI pour votre environnement. Pour plus d'informations, voir «Activation de PMI», à la page 391.
- Les chemins dans cette tâche partent du principe que vous récupérez les statistiques pour l'exemple d'application, mais vous pouvez utiliser ces statistiques pour toute autre application avec des étapes similaires.
- Si vous utilisez la console d'administration, vous devez être capable de vous y connecter. Si vous utilisez un script, vous devez être capable de vous connecter à wsadmin.

Pourquoi et quand exécuter cette tâche

Vous pouvez récupérer les statistiques PMI pour les afficher dans Tivoli Performance Viewer en suivant les étapes dans la console d'administration ou par script.

- Etapes de la console d'administration
- Etapes du script

Pour plus d'informations concernant les statistiques qui peuvent être récupérées, voir «Modules PMI», à la page 394.

Procédure

- Récupérez les statistiques PMI dans la console d'administration.
  1. Dans la console d'administration, cliquez sur **Contrôle et réglage** → **Performance viewer** → **Activité actuelle**
  2. Sélectionnez le serveur que vous voulez contrôler à l'aide de Tivoli Performance Viewer, puis activez le contrôle.
  3. Cliquez sur le serveur pour afficher la page Performance viewer.

4. Développez l'arborescence de configuration. Cliquez sur **ObjectGrid Maps** → **clusterObjectGrid**, sélectionnez **employés**. Développez **ObjectGrids** → **clusterObjectGrid** et sélectionnez **DEFAULT**.
5. Dans le modèle d'application ObjectGrid, accédez au servlet ObjectGridCreationServlet, cliquez sur le bouton 1 et remplissez les mappes. Vous pouvez afficher les statistiques dans l'afficheur.

- Récupérez les statistiques PMI avec un script.
  1. Dans une ligne de commande, accédez au répertoire `install_root/bin`. Entrez `wsadmin` pour lancer l'outil wsadmin.
  2. Définissez les variables pour l'environnement à l'aide des commandes suivantes :

     ```
     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. Définissez les variables pour obtenir les statistiques de mapModule à l'aide des commandes suivantes :

     ```
     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. Obtenez les statistiques de mapModule à l'aide de la commande suivante :

     ```
     wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params $sigs
     ```
  5. Définissez les variables pour obtenir les statistiques d'objectGridModule à l'aide des commandes suivantes :

     ```
     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. Obtenez les statistiques d'objectGridModule à l'aide de la commande suivante :

     ```
     wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params2 $sigs2
     ```

Résultats

Vous pouvez afficher les statistiques dans Tivoli Performance Viewer.

Modules PMI

Vous pouvez surveiller les performances de vos applications avec les modules PMI (Performance Monitoring Infrastructure).

objectGridModule

Le module objectGridModule contient une statistique de durée : le temps de réponse des transactions. Une transaction est définie comme la durée entre l'appel de méthode Session.begin et l'appel de méthode Session.commit. Cette durée est suivie comme temps de réponse des transactions. L'élément racine de la structure

objectGridModule, "root", sert de point d'entrée aux statistiques de WebSphere eXtreme Scale. Cet élément racine contient des ObjectGrids comme éléments enfant et ces derniers possèdent des types de transaction comme éléments enfant. Les statistiques de temps de réponse sont associées à chaque type de transaction.

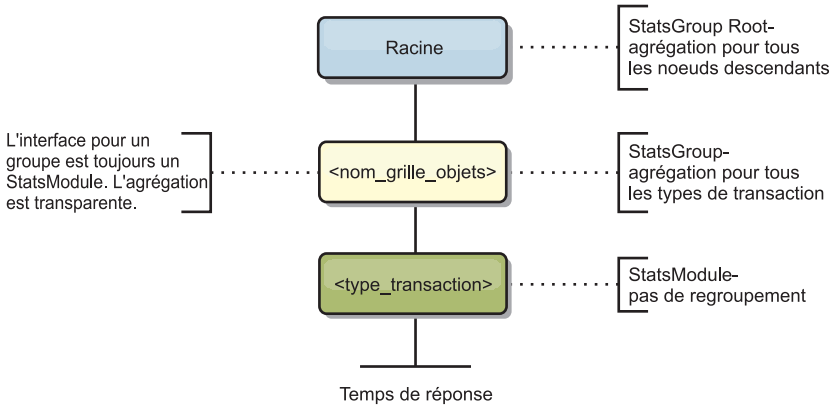


Figure 28. Structure de module ObjectGridModule

Le diagramme ci-après illustre un exemple de structure ObjectGridModule. Dans cet exemple, il existe deux instances ObjectGrid sur le système : ObjectGrid A et ObjectGrid B. L'instance ObjectGrid A possède deux types de transaction : A et default. L'instance ObjectGrid B ne possède que le type de transaction default.

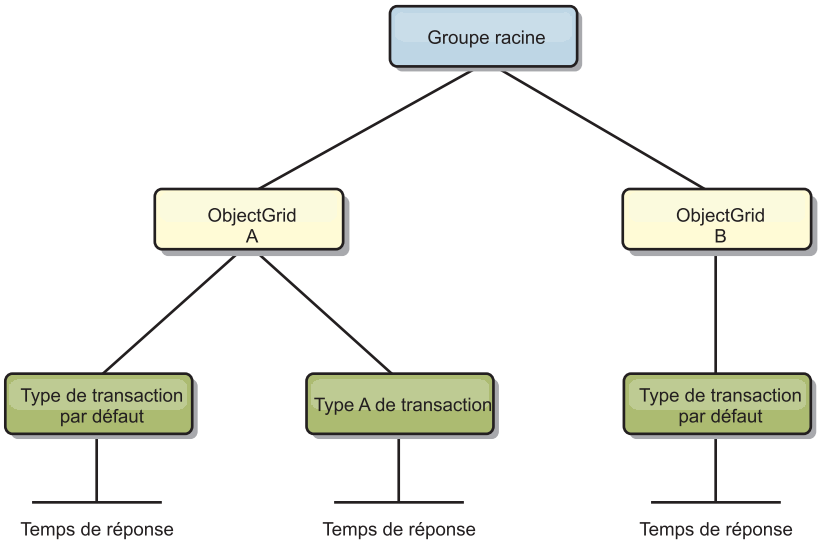


Figure 29. Exemple de structure de module ObjectGridModule

Les types de transaction sont définis par les développeurs d'applications car ils savent quels types de transaction sont utilisés par leurs applications. Le type de transaction est défini à l'aide de la méthode Session.setTransactionType(String) suivante :

```
/**
* Définit le type de transaction des transactions futures.
*
* Une fois que cette méthode a été appelée, toutes les transactions futures sont de
* même type jusqu'à ce qu'un autre type de transaction ait été défini. Si aucun type
* de transaction n'est défini, le type de transaction TRANSACTION_TYPE_DEFAULT
* par défaut est utilisé.
```

```
*
* Les types de transaction sont principalement utilisés à des fins de suivi des
* données statistiques.
* Les utilisateurs peuvent prédéfinir les types des transactions qui sont
* exécutées dans une application. L'idée consiste à regrouper les
* transactions  de mêmes caractéristiques
* dans une même catégorie (type), afin qu'une statistique de temps de réponse
* des transactions puisse être utilisée pour rechercher chaque type.
* de transaction
* Ce suivi est utile si votre application possède différents types de
* transaction.
* Parmi eux, certains types de transaction, comme les transactions de mise à
* jour, possèdent un délai de traitement supérieur à celui d'autres
* transactions, telles que les transactions en lecture seule. Si le type de
* transaction est utilisé, les différentes transactions sont recherchées par des
* statistiques différentes, afin que ces dernières puissent être plus utiles.
*
* @param tranType Type de transaction des transactions futures.
*/
void setTransactionType(String tranType);
```

L'exemple suivant spécifie `updatePrice` comme type de transaction :

```
// Spécifiez le type de transaction updatePrice
// La durée entre session.begin() et session.commit() fait l'objet d'un suivi
// dans les statistiques de durée de "updatePrice".
session.setTransactionType("updatePrice");
session.begin();
map.update(stockId, new Integer(100));
session.commit();
```

La première ligne indique que le type de transaction suivant est updatePrice. Il existe une statistiques updatePrice sous l'instance ObjectGrid qui correspond à la session de l'exemple. A l'aide d'interfaces JMX (Java Management Extensions), vous pouvez obtenir le temps de réponse des transactions updatePrice. Vous pouvez également extraire les statistiques agrégées de tous les types de transaction sur l'instance ObjectGrid spécifiée.

mapModule

La structure mapModule contient trois statistiques sur les mappes eXtreme Scale :

- **Nombre d'occurrences de mappe** - *BoundedRangeStatistic* : Recherche le nombre d'occurrences d'une mappe. Le nombre d'occurrences est une valeur flottante comprise entre 0 et 100 compris, qui représente le pourcentage d'occurrences de mappe en relation avec les opérations d'extraction de mappe.
- **Nombre d'entrées** - *CountStatistic* : Recherche le nombre d'entrées dans la mappe.
- **Temps de réponse de la mise à jour par lots du chargeur** - *TimeStatistic* : Recherche le temps de réponse utilisé pour l'opération de mise à jour par lots du chargeur.

L'élément racine de la structure mapModule, "root", sert de point d'entrée aux statistiques des mappes ObjectGrid. Cet élément racine contient des ObjectGrids comme éléments enfant et ces derniers possèdent des mappes comme éléments enfant. Trois statistiques sont répertoriées pour chaque instance de mappe. La structure mapModule est illustrée dans le diagramme suivant :

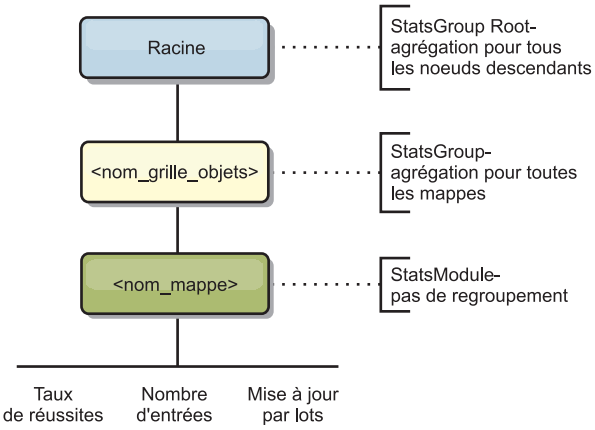
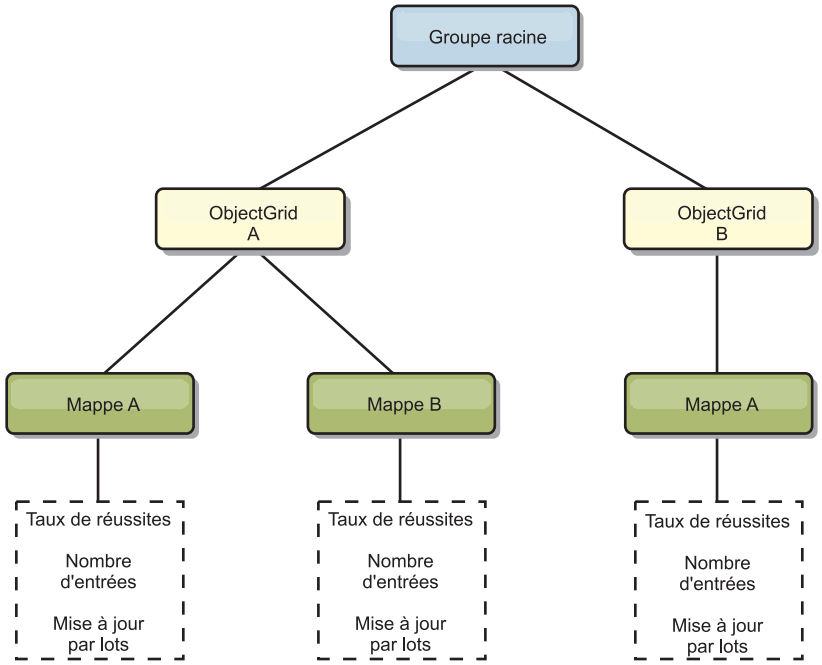


Figure 30. structure mapModule

Le diagramme suivant illustre un exemple de structure mapModule :

Figure 31. Exemple de structure de module mapModule



hashIndexModule

La structure hashIndexModule contient les statistiques suivantes sur les index de niveau mappe :

- **Nombre de recherches** - *CountStatistic* : Nombre d'appels de l'opération de recherche d'index.
- **Nombre de collisions** - *CountStatistic* : Nombre de collisions de l'opération de recherche.
- **Nombre d'échecs** - *CountStatistic* : Nombre d'échecs pour l'opération de recherche.

- **Nombre de résultats** - *CountStatistic* : Nombre de clés renvoyées par l'opération de recherche.
- **Nombre de mises à jour par lots** - *CountStatistic* : Nombre de mises à jour par lots sur cet index. Si la mappe correspondante es modifié qu'une quelconque manière, la méthode doBatchUpdate() de l'index est appelée. Cette statistiques indique la fréquence à laquelle votre index est modifié ou mis à jour.
- **Durée de recherche** - *TimeStatistic* : Temps que prend l'opération de recherche pour s'exécuter

L'élément racine de la structure hashIndexModule, "root", sert de point d'entrée aux statistiques de HashIndex. Cet élément racine contient des ObjectGrids comme éléments enfant, les ObjectGrids contiennent des mappes comme éléments enfant et enfin, ces mappes contiennent des instances HashIndex comme éléments enfant et les noeuds terminaux de l'arborescence. Trois statistiques sont répertoriées pour chaque instance HashIndex. La structure hashIndexModule est illustrée dans le diagramme suivant :

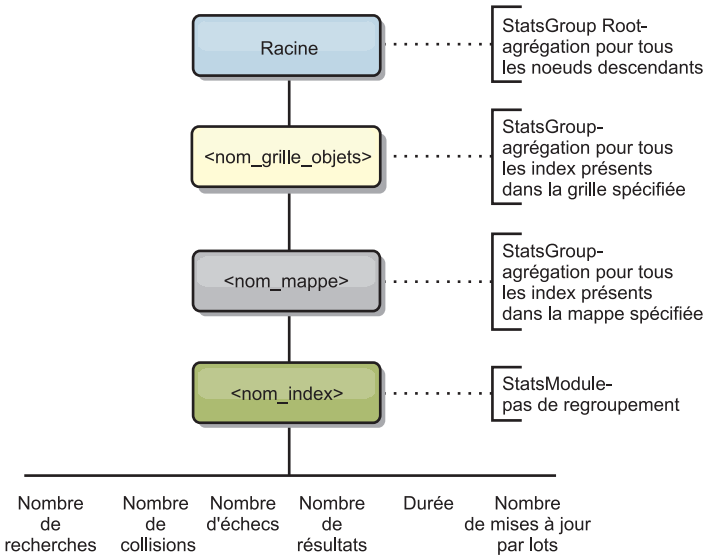


Figure 32. structure de module hashIndexModule

Le diagramme suivant illustre un exemple de structure hashIndexModule :

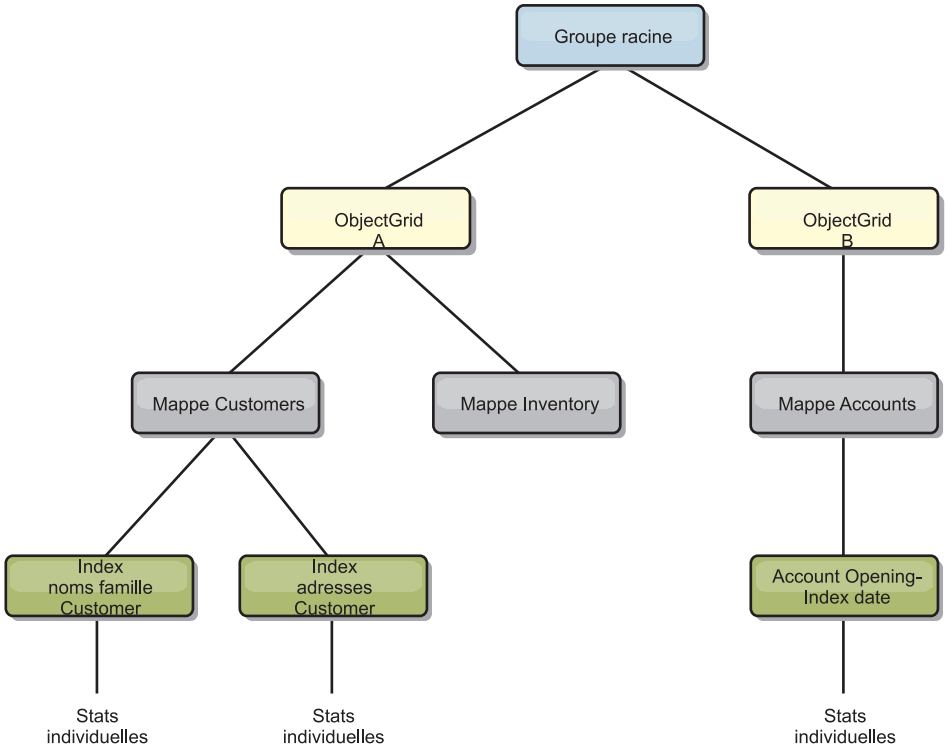


Figure 33. Exemple de structure de module hashIndexModule

agentManagerModule

La structure agentManagerModule contient les statistiques sur les agents de niveau mappe :

- **Temps de réduction** : *TimeStatistic* - Durée nécessaire pour que l'agent termine l'opération de réduction.
- **Durée totale** : *TimeStatistic* - Durée totale nécessaire à l'agent pour effectuer toutes les opérations.
- **Temps de sérialisation de l'agent** : *TimeStatistic* - Durée nécessaire pour sérialiser l'agent.
- **Temps d'inflation de l'agent** : *TimeStatistic* - Durée nécessaire pour l'inflation de l'agent sur le serveur.
- **Temps de sérialisation des résultats** : *TimeStatistic* - Durée nécessaire pour sérialiser les résultats de l'agent.
- **Temps d'inflation des résultats** : *TimeStatistic* - Durée nécessaire pour l'inflation des résultats de l'agent.
- **Nombre d'échecs** : *CountStatistic* - Nombre de fois que l'agent a échoué.
- **Nombre d'appels** : *CountStatistic* - Nombre d'appels d'AgentManager.
- **Nombre de partitions** : *CountStatistic* - Nombre de partitions vers lesquelles l'agent est envoyé.

L'élément racine de la structure agentManagerModule, "root", sert de point d'entrée aux statistiques d'AgentManager. Cet élément racine contient des ObjectGrids comme éléments enfant, les ObjectGrids contiennent des mappes comme éléments enfant et enfin, ces mappes contiennent des instances AgentManager comme éléments enfant et les noeuds terminaux de l'arborescence. Trois statistiques sont

répertoriées pour chaque instance AgentManager.

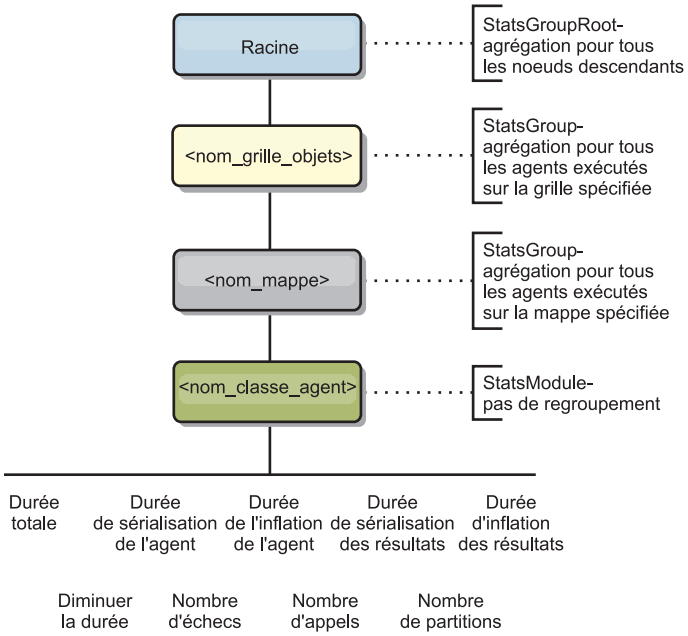


Figure 34. Structure agentManagerModule

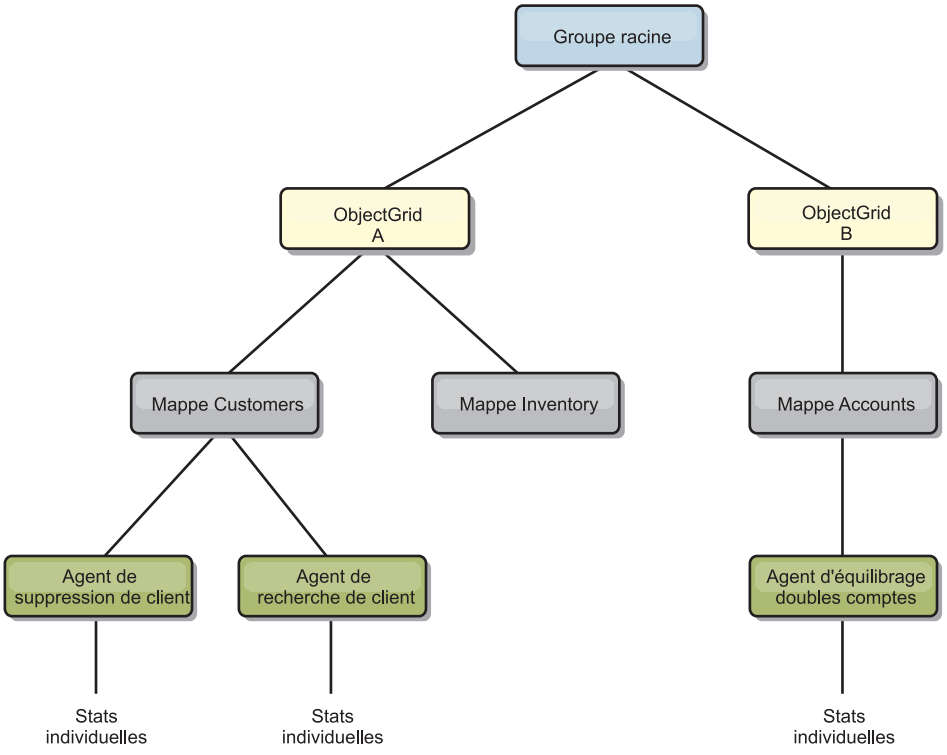


Figure 35. Exemple de structure agentManagerModule

queryModule

La structure queryModule contient les statistiques sur les requêtes eXtreme Scale :

- **Temps de création du plan** : *TimeStatistic* - Durée nécessaire pour créer le plan de requête.

- **Temps d'exécution** : *TimeStatistic* - Durée nécessaire pour exécuter la requête.
- **Nombre d'exécutions** : *CountStatistic* - Nombre de fois que la requête a été exécutée.
- **Nombre de résultats** : *CountStatistic* - Nombre de résultats pour chaque ensemble de résultats de chaque exécution de requête.
- **Nombre d'échecs** : *CountStatistic* - Nombre de fois que la requête a échoué.

L'élément racine de la structure queryModule, "root", sert de point d'entrée aux statistiques des requêtes. Cet élément racine contient des ObjectGrids comme éléments enfant et ces derniers possèdent des objets de requête comme éléments enfant et les noeuds terminaux de l'arborescence. Trois statistiques sont répertoriées pour chaque instance de requête.

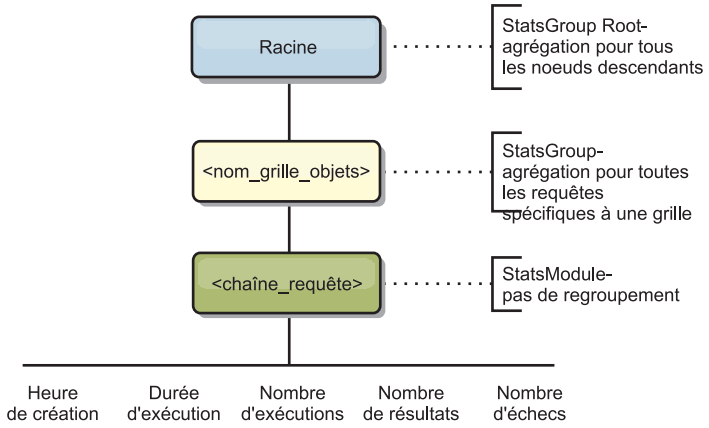


Figure 36. structure queryModule

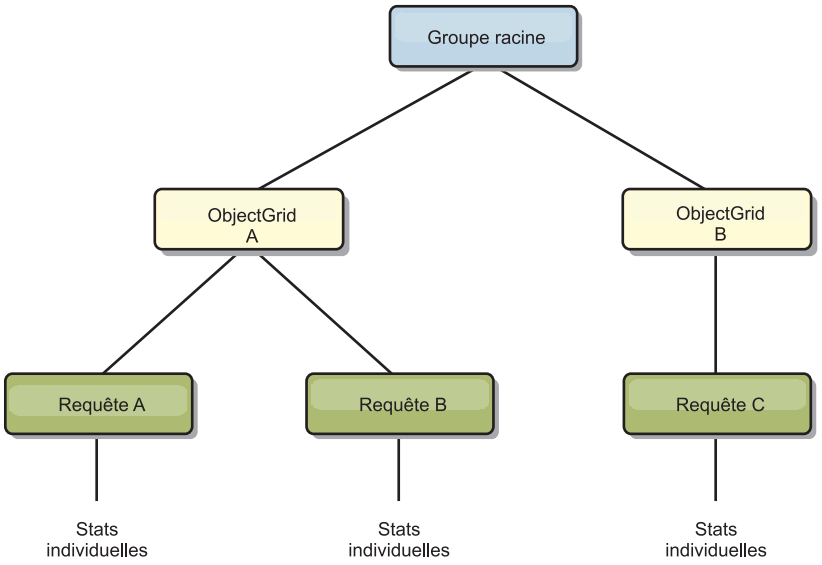


Figure 37. Exemple de structure queryModule QueryStats.jpg

Accès aux beans gérés à l'aide de l'outil wsadmin

Vous pouvez utiliser l'utilitaire wsadmin fourni dans WebSphere Application Server pour accéder aux informations sur les beans gérés.

Exécutez l'outil wsadmin depuis le répertoire bin de votre installation WebSphere Application Server. L'exemple suivant restaure une vue de la position actuelle du fragment dans un logiciel eXtreme Scale dynamique. Vous pouvez exécuter wsadmin depuis n'importe quelle installation où eXtreme Scale est en cours d'exécution. Vous n'avez pas besoin de l'exécuter sur le service de catalogue.

```
$ 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>
```

Surveillance à l'aide de beans gérés (MBeans)

Vous pouvez utiliser des beans gérés (MBeans) pour effectuer le suivi des statistiques dans votre environnement.

Avant de commencer

Pour que les attributs puissent être enregistrés, vous devez activer les statistiques. Vous pouvez activer les statistiques de l'une des manières suivantes :

- **A l'aide du fichier de propriétés du serveur :**

  Vous pouvez activer les statistiques dans le fichier de propriétés du serveur avec l'entrée clé-valeur `statsSpec=<SpécStats>`. Voici quelques exemples de paramètres possibles :

  – pour activer toutes les statistiques, utilisez `statsSpec=all=enabled`

  – pour n'activer que les statistiques d'ObjectGrid, utilisez `statsSpec=og.all=enabled` Pour une description de toutes les spécifications de statistiques possibles, voir l'API StatsSpec dans la documentation de l'API.

  Pour plus d'informations sur le fichier de propriétés du serveur, voir «Fichier de propriétés du serveur», à la page 185.

- **Avec un bean géré :**

  Les statistiques peuvent désormas être activées à l'aide de l'attribut StatsSpec du bean géré ObjectGrid. Pour des informations complémentaires, voir API StatsSpec.

- **Par programmation :**

  Vous pouvez également programmer l'activation des statistiques avec l'interface StatsAccessor, qui est extraite avec la classe StatsAccessorFactory. Utilisez cette interface dans un environnement client ou lorsque vous devez surveiller une instance d'eXtreme Scale exécutée dans le processus en cours.

Exemple

Pour obtenir un exemple d'utilisation des beans gérés, voir «Surveillance à l'aide de l'exemple d'utilitaire xsAdmin», à la page 387.

Surveillance à l'aide de la console Web

L'une des nouvelles fonctionnalités d'eXtreme Scale 7.1 est la console Web qui permet de représenter graphiquement les statistiques actuelles ou historiques. Cette console fournit en effet un certain nombre de graphiques prédéfinis pour une vision d'ensemble générale et elle comporte une page de rapports personnalisés que vous pouvez utiliser pour élaborer des graphiques à partir des statistiques disponibles. Les fonctionnalités graphiques de la console de surveillance de WebSphere eXtreme Scale permettent de visualiser les performances globales des grilles des données présentes dans votre environnement.

Avant de commencer

Le serveur de la console s'exécute sous AIX, Linux ou Windows. Le système du serveur de la console doit être capable de se connecter à votre service de catalogue tout comme ce dernier doit lui aussi être capable de se connecter en retour au serveur de la console. Vous avez besoin d'une installation autonome de WebSphere eXtreme Scale sur le système qui va héberger le serveur de la console. Le script startConsoleServer.bat|sh de démarrage du serveur de la console est situé dans le répertoire XS_ROOT/ObjectGrid/bin de votre installation.

Une fois les grilles de données créées et les applications configurées pour utiliser ces grilles, laissez aux statistiques le temps d'être générées. Par exemple, avec une grille de données de caches dynamiques, les statistiques ne seront utilisables qu'une fois qu'un WebSphere Application Server exécutant un cache dynamique se sera connecté à cette grille. Si vous utilisez une collectivité, l'initialisation de cette collectivité doit être terminée pour que les statistiques soient disponibles. En général, il suffit d'attendre environ une minute après l'apport d'une modification de configuration importante pour observer le changement au niveau des statistiques.

Conseil : Il suffit de déplacer le pointeur de la souris sur n'importe quel point de données du graphique pour afficher des informations plus précises sur ce point.

Procédure

- Démarrez le serveur de la console. Le script startConsoleServer.bat|sh de démarrage du serveur de la console est situé dans le répertoire XS_ROOT/ObjectGrid/bin de votre installation.
- Ouvrez une session sur la console.
  1. Dans votre navigateur Web, allez à `https://hôte.votre.console:7443` en remplaçant `hôte.votre.console` par le nom d'hôte de la machine sur laquelle est installée la console.
  2. Ouvrez une session sur la console.
     - **ID utilisateur :** `admin`
     - **Mot de passe :** `admin`

     La page d'accueil de la console s'affiche.
  3. Cliquez sur **Paramètres > Configuration** pour afficher la configuration de la console. La configuration de la console comprend ce type d'informations :

- la chaîne de trace pour le client WebSphere eXtreme Scale, comme `*=all=disabled`
- le nom et le mot de passe de l'administrateur
- son adresse e-mail

- Visualisez le statut de la connexion
  1. Allez à la page d'accueil en cliquant sur le lien **Accueil** dans la barre d'outils.
  2. A droite de la barre d'outils figure une liste déroulante qui affiche le domaine auquel est connecté le serveur de la console. A droite de cette liste déroulante figure un indicateur de statut de la connexion.
- Créez et maintenez des connexions aux serveurs de catalogue que vous voulez surveiller.
  1. Allez à la page **Paramètres > Serveurs de catalogue eXtreme Scale**.
  2. Ajoutez un nouveau serveur de catalogue.
     a. Cliquez sur le signe Plus pour afficher une boîte de dialogue permettant d'enregistrer un serveur de catalogue existant.
     b. Entrez les informations requises (nom d'hôte, port JMX et port d'écoute).

        Nom d'hôte

        Affiche le nom d'hôte du poste de travail sur lequel s'exécute le service de catalogue.

        Port JMX

        Affiche le numéro du port par l'intermédiaire duquel les connexions JMX/RMI sont activées. JMX permet la surveillance et la gestion à partir de systèmes éloignés.

        Port d'écoute

        Indique le port d'écoute pour les communications avec le protocole IIOP (Internet Inter-ORB Protocol).

     c. Cliquez sur **OK**.
     d. Vérifiez que le serveur de catalogue a bien été ajouté à l'arborescence de navigation.

  Pour afficher les informations concernant un serveur de catalogue existant, dans la page **Paramètres > Serveurs de catalogue eXtreme Scale**, cliquez sur le nom de ce serveur dans l'arborescence de navigation.
- Créez et maintenez des connexions aux domaines que vous voulez surveiller.
  1. Allez à la page **Paramètres > Domaines eXtreme Scale**.
  2. Ajoutez un nouveau domaine.
     a. Cliquez sur le signe Plus pour afficher une boîte de dialogue permettant d'enregistrer un domaine de services de catalogue.
     b. Fournissez les informations requises.

        Nom Affiche le nom d'hôte du domaine, celui attribué par l'administrateur.

        Utilisateur JMX
        Affiche le nom d'utilisateur JMX (Java Management Extensions) à employer (si la sécurité est activée).

        Mot de passe JMX
        Affiche le mot de passe JMX si la sécurité est activée.

Serveurs de catalogue
Affiche la liste du ou des serveurs de catalogue appartenant au domaine sélectionné.

Classe de génération
Affiche le nom de la classe qui implémente l'interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Cette classe est utilisée pour obtenir les données d'identification des clients.

Propriétés du générateur
Affiche les propriétés de la classe d'implémentation CredentialGenerator. Les propriétés sont définies pour l'objet avec la méthode setProperties(String). La valeur credentialGeneratorprops n'est utilisée que si la valeur de la propriété credentialGeneratorClass n'est pas null.

c. Cliquez sur OK.

d. Vérifiez que le domaine a bien été ajouté à l'arborescence de navigation.

3. Spécifiez dans cette page les informations de sécurité requises.

4. Associez le domaine aux serveurs de catalogue que vous avez précédemment ajoutés.

Pour afficher les informations concernant un domaine existant, dans la page **Paramètres > Domaines eXtreme Scale**, cliquez sur le nom de ce domaine dans l'arborescence de navigation.

- Pour afficher les statistiques en cours du serveur, cliquez sur **Surveiller > Vue d'ensemble du serveur**.

  Statistiques du serveur

  Mémoire utilisée
  Affiche la quantité de mémoire actuellement utilisée (mémoire réelle) dans l'environnement d'exécution du serveur. Ne sont affichés que les 25 premiers serveurs utilisant le plus de mémoire.

  Total de la mémoire utilisée dans le temps
  Affiche l'utilisation de la mémoire réelle dans l'environnement d'exécution du serveur.

  Mémoire utilisée au fil du temps
  Affiche la quantité de mémoire utilisée dans l'environnement d'exécution du serveur.

- Pour afficher les performances de toutes vos grilles de données, cliquez sur **Surveiller > Vue d'ensemble des domaines de grilles de données**.

  Statistiques générales du domaine de grilles des données

  Répartition des capacités actuelles utilisées par les grilles de données
  Ce graphique contient une image du pool total et des plus grands consommateurs de capacités. Ne sont affichées que les 25 grilles de données les plus consommatrices.

  Cinq premières grilles de données par durée moyenne de transaction en millisecondes
  Ce graphique contient la liste des cinq premiers caches de données, classés par durée moyenne de transaction.

Cinq premières grilles de données par débit moyen en transaction par seconde

Ce graphique contient la liste des cinq premières grilles des données, classées par débit moyen en transactions par seconde.

- Pour afficher les performances de grilles prises individuellement, cliquez sur **Surveiller > Vue d'ensemble des domaines de grilles de données>** *nom_grille*. Cette page propose un récapitulatif incluant le nombre d'entrées en cache, la durée moyenne des transactions et le débit moyen. Vous pouvez également afficher les graphiques suivants :

  Statistiques générales des grilles des données

  Récapitulatif courant

  Affiche des statistiques du type nombre actuel d'objets de cette grille mis en cache (entrées du cache), durée moyenne des transactions et débit moyen des transactions.

  Capacités utilisées vs. Nombre d'entrées en cache

  Ce graphique montre les capacités utilisées du cache par rapport au nombre d'entrées dans ce dernier. Vous pouvez modifier l'intervalle affiché : dernière heure, dernier jour, dernière semaine, dernier mois. Le niveau de détail affiché dans le graphique varie en fonction de l'intervalle sélectionné.

  Nombre total de demandes d'extraction en cache vs. Demandes d'extraction en cache ayant abouti

  Ce graphique permet de visualiser le nombre de requêtes ayant réussi dans le cache.

- Pour afficher d'autres détails sur une grille de données spécifique, cliquez sur **Surveiller > Détails d'une grille de données**. Une arborescence affiche toutes les grilles de données de votre configuration. Vous pouvez explorer cette arborescence en aval et accéder à une grille de données spécifique afin d'afficher les mappes appartenant à cette grille. Vous pouvez cliquer sur le nom d'une grille de données ou d'une mappe pour obtenir plus d'informations.

  Statistiques d'une grille de données

  Récapitulatif courant

  Permet de visualiser la capacité actuelle utilisée ainsi que la liste des zones auxquelles appartient la grille de données.

  Répartition des capacités actuelles utilisées par les mappes de grilles d'objets eXtreme Scale

  Permet de visualiser un pool total, ce qui inclut les capacités par zone ainsi que les capacités totales dans chaque zone. Ne sont affichées que les 25 plus importantes mappes ObjectGrid.

  Répartition des capacités actuelles utilisées par les zones

  Permet d'afficher une zone, ce qui inclut le pool total ainsi que les principaux consommateurs des capacités utilisées. Ne sont affichées que les 25 zones les plus consommatrices.

  Statistiques d'une mappe

  Récapitulatif courant

  Afficher de statistiques du type :

  Capacité utilisée

  Permet de visualiser la capacité utilisée ainsi que la liste des zones auxquelles appartient la mappe.

Nombre d'entrées du cache
Affiche le nombre d'objets de la mappe mis en cache.

Délai de transaction moyen (ms)
Affiche la durée moyenne d'exécution des transactions impliquant cette mappe.

Débit de transaction moyen (trans/sec)
Affiche le nombre moyen de transactions par seconde impliquant cette mappe.

Répartition des capacités actuelles utilisées par les partitions
Ce graphique contient une image du pool total ainsi que des plus gros consommateurs de capacités utilisées. Ne sont affichées que les 25 partitions les plus consommatrices.

- Pour choisir les statistiques que vous voulez voir figurer dans vos rapports personnalisés, cliquez sur **Surveiller > Rapports personnalisés**.

  Cette vue permet d'élaborer des graphiques détaillés à partir des diverses statistiques. L'arborescence permet d'explorer les grilles de données et les serveurs disponibles, ainsi que leurs statistiques. Un menu s'affiche lorsqu'on clique ou que l'on appuie sur Entrée sur un noeud qui référence des données pouvant être représentées dans un graphique. Vous pouvez créer un nouveau graphique contenant les statistiques ou, si elles sont compatibles, ajouter ces statistiques à celles d'un graphique existant.

  Statistiques de domaine

  Délai de transaction moyen (ms)
  Affiche la durée moyenne que met une transaction à s'effectuer dans ce domaine.

  Débit de transaction moyen (trans/sec)
  Affiche le nombre moyen de transactions par seconde dans ce domaine.

  Délai maximum de transaction (ms)
  Affiche le temps *maximum* qu'a mis une transaction pour s'exécuter dans ce domaine.

  Délai minimum de transaction (ms)
  Affiche le temps *minimum* qu'a mis une transaction pour s'exécuter dans ce domaine.

  Délai de transaction total (ms)
  Affiche le temps total passé à des transactions dans ce domaine depuis l'initialisation de ce dernier.

  Statistiques de conteneur eXtreme Scale

  Délai de transaction moyen (ms)
  Affiche pour ce serveur de catalogue la durée moyenne que met une transaction à s'effectuer.

  Débit de transaction moyen (trans/sec)
  Affiche le nombre moyen de transactions par seconde pour ce serveur de catalogue.

  Délai maximum de transaction (ms)
  Affiche pour ce serveur de catalogue le temps *maximum* qu'a mis une transaction pour s'exécuter.

  Délai minimum de transaction (ms)
  Affiche pour ce serveur de catalogue le temps *minimum* qu'a mis une transaction pour s'exécuter.

Délai de transaction total (ms)
Affiche pour ce serveur de catalogue le temps total passé à des transactions depuis l'initialisation du serveur.

Nombre total d'entrées en cache
Affiche le nombre actuel d'objets mis en cache appartenant à des grilles supervisées par ce serveur de catalogue.

Nombre maximal d'entrées en cache
Affiche le nombre maximum d'objets mis en cache appartenant à des grilles supervisées par ce serveur de catalogue.

Nombre minimal d'entrées en cache
Affiche le nombre minimum d'objets mis en cache appartenant à des grilles supervisées par ce serveur de catalogue.

Taux de réussites (pourcentage)
Affiche le taux de réussites pour la grille de données sélectionnée. Un taux élevé est souhaitable. Ce taux indique le degré d'efficacité de la grille pour éviter d'accéder au stockage de persistance.

Octets utilisés
Affiche la consommation de la mémoire par cette mappe.

Nombre minimal d'octets utilisés
Affiche le point bas de la consommation de mémoire par ce service de catalogue et ses mappes.

Nombre maximal d'octets utilisés
Affiche le point haut de la consommation de mémoire par ce service de catalogue et ses mappes.

Nombre total de réussites
Affiche le nombre total de fois où les données demandées ont été trouvées dans la mappe, dispensant de devoir accéder au stockage de persistance.

Nombre total de demandes get
Affiche le nombre total de fois où la mappe a dû accéder au stockage de persistance pour obtenir des données.

Segments de mémoire disponibles (Mo)
Affiche la quantité effective de segments mémoire disponibles pour la machine virtuelle Java en cours d'utilisation par le serveur de catalogue.

Total des segments de mémoire
Affiche la quantité effective de segments mémoire disponibles pour la machine virtuelle Java en cours d'utilisation par ce serveur de catalogue.

Mémoire utilisée
Affiche la mémoire utilisée dans la machine virtuelle Java en cours d'utilisation par ce serveur de catalogue.

Nombre de processeurs disponibles
Affiche le nombre de processeurs disponibles pour ce service de catalogue et ses mappes. Pour une stabilité optimale, vous devez exécuter les serveurs à 60 % du chargement de processeur et les segments de mémoire des machines virtuelles Java à 60 % du chargement des segments de mémoire. Les pics peuvent alors pousser l'utilisation du processeur à 80-90 %, mais ce ne doit pas être le niveau habituel d'exécution de vos serveurs.

Taille maximale des segments de mémoire (Mo)
Affiche la quantité maximale de segments mémoire disponibles pour la machine virtuelle Java en cours d'utilisation par ce serveur de catalogue.

Statistiques de grille

Délai de transaction moyen (ms)
Affiche la durée moyenne que mettent pour s'effectuer des transactions impliquant cette grille.

Débit de transaction moyen (trans/sec)
Affiche le nombre moyen de transactions effectuées par seconde par cette grille.

Délai maximum de transaction (ms)
Affiche le temps *maximum* qu'a mis une transaction effectuée par cette grille.

Délai minimum de transaction (ms)
Affiche le temps *minimum* qu'a mis une transaction effectuée par cette grille.

Délai de transaction total (ms)
Affiche le temps total de traitement des transactions pour cette grille.

Statistiques d'une mappe

Nombre total d'entrées en cache
Affiche le nombre d'objets de la mappe actuellement mis en cache.

Nombre maximal d'entrées en cache
Affiche le nombre maximum d'objets mis en cache appartenant à cette mappe depuis l'initialisation de cette dernière.

Nombre minimal d'entrées en cache
Affiche le nombre minimum d'objets mis en cache appartenant à cette mappe depuis l'initialisation de cette dernière.

Taux de réussites (pourcentage)
Affiche le taux de réussites pour la mappe sélectionnée. Un taux élevé est souhaitable. Ce taux indique le degré d'efficacité de la mappe pour éviter d'accéder au stockage de persistance.

Octets utilisés
Affiche la consommation de la mémoire par cette mappe.

Nombre minimal d'octets utilisés
Affiche la consommation minimum en octets pour cette mappe.

Nombre maximal d'octets utilisés
Affiche la consommation maximum en octets pour cette mappe.

Nombre total de réussites
Affiche le nombre total de fois où les données demandées ont été trouvées dans la mappe, dispensant de devoir accéder au stockage de persistance.

Nombre total de demandes get
Affiche le nombre total de fois où la mappe a dû accéder au stockage de persistance pour obtenir des données.

Segments de mémoire disponibles (Mo)
Affiche la quantité effective de segments mémoire disponibles pour cette mappe dans la machine virtuelle Java en cours d'utilisation par le serveur de catalogue.

Total des segments de mémoire (Mo)
Affiche la quantité totale de segments mémoire disponibles pour cette mappe dans la machine virtuelle Java en cours d'utilisation par le serveur de catalogue. Pour une stabilité optimale, vous devez exécuter les serveurs à 60 % du chargement de processeur et les segments de mémoire des machines virtuelles Java à 60 % du chargement des segments de mémoire. Les pics peuvent alors pousser l'utilisation du processeur à 80-90 %, mais ce ne doit pas être le niveau habituel d'exécution de vos serveurs.

Mémoire utilisée (Mo)
Affiche la quantité de mémoire utilisée dans cette mappe.

Nombre de processeurs disponibles
Affiche le nombre de processeurs disponibles pour cette mappe. Pour une stabilité optimale, vous devez exécuter les serveurs à 60 % du chargement de processeur et les segments de mémoire des machines virtuelles Java à 60 % du chargement des segments de mémoire. Les pics peuvent alors pousser l'utilisation du processeur à 80-90 %, mais ce ne doit pas être le niveau habituel d'exécution de vos serveurs.

Taille maximale des segments de mémoire (Mo)
Affiche la quantité maximum de segments mémoire disponibles pour cette mappe dans la machine virtuelle Java en cours d'utilisation par le serveur de catalogue.

Surveillance à l'aide d'outils fournis par une tierce partie

WebSphere eXtreme Scale peut être surveillé à l'aide de plusieurs solutions de surveillance d'entreprise couramment utilisées. Des agents de plug-in sont intégrés pour IBM Tivoli Monitoring et Hyperic HQ dont le rôle consiste à surveiller WebSphere eXtreme Scale à l'aide de beans de gestion accessibles publiquement. CA Wily Introscope utilise l'instrumentation de méthode Java pour capturer les statistiques.

Surveillance à l'aide d'IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale

L'agent IBM Tivoli Enterprise Monitoring est une solution de surveillance riche en fonctions que vous pouvez utiliser pour surveiller les bases de données, les systèmes d'exploitation et les serveurs dans des environnements hôte et répartis. WebSphere eXtreme Scale inclut un agent personnalisé que vous pouvez utiliser pour introspecter les beans de gestion d'eXtreme Scale. Cette solution fonctionne correctement pour les déploiements eXtreme Scale autonomes et les déploiements WebSphere Application Server.

Avant de commencer

- Installez WebSphere eXtreme Scale Version 7.0.0 ou ultérieure.

  Par ailleurs, les statistiques doivent être activées pour permettre la collecte de données statistiques à partir des serveurs WebSphere eXtreme Scale. Les diverses options d'activation des statistiques sont décrites dans «Surveillance à l'aide de beans gérés (MBeans)», à la page 402 et dans «Surveillance à l'aide de l'exemple d'utilitaire xsAdmin», à la page 387.
- Installez IBM Tivoli Monitoring Version 6.2.1 avec le Fix Pack 2 ou ultérieur.
- Installez l'agent du système d'exploitation Tivoli sur chaque serveur ou hôte sur lequel des serveurs eXtreme Scale sont exécutés.

- Installez l'agent WebSphere eXtreme Scale, que vous pouvez télécharger gratuitement à partir du site IBM Open Process Automation Library (OPAL).

Effectuez les étapes suivantes pour installer et configurer Tivoli Monitoring Agent :

Procédure

1. Installez Tivoli Monitoring Agent for WebSphere eXtreme Scale.

   Téléchargez l'image d'installation de Tivoli et extrayez ses fichiers dans un répertoire temporaire.

2. Installez les fichiers du support d'application d'eXtreme Scale.

   Installez le support d'application d'eXtreme Scale sur chacun des déploiements ci-après.

   - Tivoli Enterprise Portal Server (TEPS)
   - Client Enterprise Desktop (TEPD)
   - Tivoli Enterprise Monitoring Server (TEMS)

   a. Dans le répertoire temporaire que vous avez créé, démarrez une nouvelle fenêtre de commande et exécutez le fichier exécutable approprié pour votre plateforme. Le script d'installation détecte automatiquement votre type de déploiement Tivoli (TEMS, TEPD ou TEPS). Vous pouvez installer tout type de déploiement sur un ou plusieurs hôtes ; ces trois types de déploiement requièrent l'installation des fichiers du support d'application de l'agent eXtreme Scale.

   b. Dans la fenêtre **Programme d'installation**, vérifiez que la sélection des composants Tivoli déployés est correcte. Cliquez sur **Suivant**.

   c. Si vous y êtes invité, soumettez votre nom d'hôte et vos données d'identification administratives. Cliquez sur **Suivant**.

   d. Sélectionnez **Monitoring Agent for WebSphere eXtreme Scale**. Cliquez sur **Suivant**.

   e. Le système vous indique les actions d'installation à effectuer. Cliquez sur **Suivant** ; vous pouvez voir la progression de l'installation jusqu'à sa fin.

   Une fois que vous avez terminé cette procédure, tous les fichiers du support d'application requis par l'agent WebSphere eXtreme Scale sont installés.

3. Installez l'agent sur chacun des noeuds eXtreme Scale.

   Vous installez un agent de système d'exploitation Tivoli sur chacun des ordinateurs. Vous n'avez pas besoin de configurer ou de démarrer cet agent. Utilisez l'image d'installation de l'étape précédente pour exécuter le fichier exécutable spécifique à la plateforme.

   Vous n'avez besoin d'installer qu'un seul agent par hôte. Chaque agent peut prendre en charge plusieurs instances de serveur eXtreme Scale. Pour de meilleures performances, utilisez une instance d'agent pour surveiller environ 50 serveurs eXtreme Scale.

   a. Dans l'écran de bienvenue de l'assistant d'installation, cliquez sur **Suivant** pour ouvrir l'écran et spécifier les informations sur le chemin d'installation.

   b. Dans la zone **Répertoire d'installation d'IBM Tivoli Monitoring**, entrez ou recherchez `C:\IBM\ITM` (ou `/opt/IBM/ITM`). Ensuite, dans la zone de l'**emplacement du support installable**, vérifiez que la valeur affichée est correcte et cliquez sur **Suivant**.

   c. Sélectionnez les composants à ajouter, tels que **Effectuer une installation locale de la solution**, et cliquez sur **Suivant**.

d. Sélectionnez les applications pour lesquelles vous souhaitez ajouter le support en les sélectionnant (par exemple, **Monitoring Agent for WebSphere eXtreme Scale**), puis en cliquant sur **Suivant**.

e. La progression s'affiche jusqu'à ce que la prise en charge de l'application ait été ajoutée.

Remarque : Répétez ces étapes sur chacun des noeuds eXtreme Scale. Vous pouvez également utiliser une installation en mode silencieux. Pour plus d'informations sur l'installation en mode silencieux, voir le Centre de documentation d'IBM Tivoli Monitoring.

4. Configurez l'agent WebSphere eXtreme Scale.

   Chacun des agents installés doit être configuré pour surveiller un serveur de catalogues et/ou un serveur eXtreme Scale.

   Les étapes permettant de configurer les plateformes Windows et UNIX sont différentes. La configuration pour la plateforme Windows est effectuée à l'aide de l'interface graphique **Gérer les services Tivoli Monitoring**. La configuration des plateformes UNIX est effectuée par l'intermédiaire de la ligne de commande.

   Windows Utilisez les étapes suivantes pour configurer initialement l'agent sous Windows

   a. Dans la fenêtre **Gérer les services Tivoli Enterprise Monitoring**, cliquez sur **Démarrrer → Tous les programmes → IBM Tivoli Monitoring → Gérer les services Tivoli Monitoring**.

   b. Cliquez à l'aide du bouton droit de la souris sur **Monitoring Agent for WebSphere eXtreme Scale** et sélectionnez **Configure using default**, qui ouvre une fenêtre permettant de créer une instance unique de l'agent.

   c. Choisissez un nom unique (par exemple, `instance1` et cliquez sur **Suivant**).

   • Si vous prévoyez de surveiller des serveurs eXtreme Scale autonomes, effectuez les étapes suivantes :

      a. Mettez à jour les paramètres Java et assurez-vous que la valeur **Java Home** est correcte. Les arguments JVM peuvent rester vides. Cliquez sur **Suivant**.

      b. Sélectionnez le type **Type de connexion de serveur MBean** et utilisez `Serveur conforme à JSR-160` pour les serveurs eXtreme Scale autonomes. Cliquez sur **Suivant**.

      c. Si la sécurité est activée, mettez à jour les valeurs **ID utilisateur** et **Mot de passe**. Ne modifiez pas la valeur d'**URL de service JMX**. Vous remplacerez cette valeur ultérieurement. Ne modifiez pas la zone **Informations de chemin de classes JMX**. Cliquez sur **Suivant**.

      Pour configurer les serveurs de l'agent sous Windows, procédez comme suit :

      a. Configurez des instances de sous-noeud des serveurs eXtreme Scale dans la sous-fenêtre **Serveurs de grille WebSphere eXtreme Scale**. S'il n'existe pas de serveurs de catalogues sur votre ordinateur, cliquez sur **Suivant** pour passer à la sous-fenêtre du service de catalogue.

      b. S'il existe plusieurs serveurs conteneurs eXtreme Scale sur votre ordinateur, configurez l'agent pour qu'il surveille chacun d'eux.

      c. Vous pouvez ajouter autant de serveurs eXtreme Scale que nécessaire, si leurs noms et ports sont uniques, en cliquant sur **Nouveau**. (Si un serveur eXtreme Scale est démarré, une valeur JMXPort doit être spécifiée.)

d. Une fois que vous avez configuré les serveurs conteneurs, cliquez sur **Suivant** pour accéder à la sous-fenêtre **Serveurs de catalogues WebSphere eXtreme Scale**.

e. En l'absence de serveurs de catalogues, cliquez sur **OK**. Si vous possédez des serveurs de catalogues, ajoutez une nouvelle configuration pour chaque serveur, comme pour les serveurs conteneurs. Choisissez de nouveau un nom unique, de préférence, celui utilisé au démarrage du service de catalogue. Cliquez sur **OK** pour terminer.

- Si vous prévoyez de surveiller les serveurs de l'agent sur des serveurs eXtreme Scale imbriqués dans un processus WebSphere Application Server, procédez comme suit :

  a. Mettez à jour les paramètres Java et assurez-vous que la valeur **Java Home** est correcte. Les arguments JVM peuvent rester vides. Cliquez sur **Suivant**.

  b. Sélectionnez le **type de connexion du serveur MBean**. Sélectionnez la version WebSphere Application Server qui convient pour votre environnement. Cliquez sur **Suivant**.

  c. Vérifiez que les informations WebSphere Application Server du panneau sont correctes. Cliquez sur **Suivant**.

  d. N'ajoutez qu'une définition de sous-noeud. Nommez cette définition de sous-noeud, mais ne mettez pas à jour la définition du port. Dans un environnement WebSphere Application Server, les données peuvent être collectées sur tous les serveurs d'applications gérés par l'agent de noeud exécuté sur l'ordinateur. Cliquez sur **Suivant**.

  e. S'il n'existe pas de serveurs de catalogues dans l'environnement, cliquez sur **OK**. S'il en existe, ajoutez une nouvelle configuration pour chaque serveur de catalogues, comme pour les serveurs conteneurs. Choisissez un nom unique pour le service de catalogue, de préférence, celui que vous avez utilisé au démarrage du service de catalogue. Cliquez sur **OK** pour terminer.

Remarque : Les serveurs conteneurs n'ont pas besoin d'être regroupés avec le service de catalogue.

Maintenant que l'agent et les serveurs sont configurés et prêts, dans la fenêtre qui suit, cliquez à l'aide du bouton droit de la souris sur `instance1` pour démarrer l'agent.

UNIX Pour configurer l'agent sur la plateforme UNIX, sur la ligne de commande, procédez comme suit :

Un exemple est illustré ci-après pour les serveurs autonomes qui utilisent un type de connexion compatible JSR160. Cet exemple illustre trois conteneurs eXtreme Scale sur l'hôte unique (rhea00b02) et les adresses du programme d'écoute JMX sont respectivement 15000, 15001 et 15002. Il n'existe pas de serveurs de catalogues.

La sortie de l'utilitaire de configuration est affichée en *`italiques à espacement fixe`*, tandis que la réponse de l'utilisateur est en **`gras à espacement fixe`**. (Si aucune réponse utilisateur n'est requise, la valeur par défaut est sélectionnée en appuyant sur la touche Entrée.)

```
rhea00b02 # ./itmcmd config -A xt
Configuration de l'agent démarrée...
Entrez un nom d'instance (la valeur par défaut est : ) : inst1
Modifier les paramètres "Monitoring Agent for WebSphere eXtreme Scale" ? [ 1=Oui, 2=Non ] (la sélection par défaut est 1) :
Modifier les paramètres 'Java' ? [ 1=Oui, 2=Non ] (la sélection par défaut est 1) :
Répertoire de base Java (la sélection par défaut est C:\Program Files\IBM\Java50) : /opt/OG61/java
Niveau de trace Java [ 1=Erreur, 2=Avertissement, 3=Informations, 4=Débogage minimum, 5=Débogage moyen, 6=Débogage maximum,
    7=Tous ] (la sélection par défaut est 1) :
Arguments JVM (la sélection par défaut est ):
Modifier les paramètres 'Connexion' ? [ 1=Oui, 2=Non ] (la sélection par défaut est 1) :
Type de connexion de serveur MBean [ 1=Serveur conforme à JSR-160, 2=WebSphere Application Server version 6.0,
3=WebSphere Application Server version 6.1, 4=WebSphere Application Server version 7.0 ] (la sélection par défaut est 1) : 1
Modifier les paramètres 'Serveur conforme à JSR-160' ? [ 1=Oui, 2=Non ] (la sélection par défaut est 1) :
ID utilisateur JMX (la sélection par défaut est ) :
Entrez Mot de passe JMX (la sélection par défaut est ) :
Entrez de nouveau : Mot de passe JMX (la sélection par défaut est ) :
Adresse URL de service JMX (la valeur par défaut est : service:jmx:rmi:///jndi/rmi://localhost:port/objectgrid/MBeanServer) :
----------------------------------------
Informations de chemin de classe JMX
Chemins de base JMX (la sélection par défaut est ) :
Chemin de classes JMX (la sélection par défaut est ) :
Répertoires JAR JMX  (la sélection par défaut est ) :
Modifier les paramètres 'Service de catalogue WebSphere eXtreme Scale' ? [ 1=Oui, 2=Non ] (la sélection par défaut est 1) : 2
Modifier les paramètres 'Serveurs de grille WebSphere eXtreme Scale' ? [ 1=Oui, 2=Non ] (la sélection par défaut est 1) : 1
Aucun paramètre 'Serveurs de grille WebSphere eXtreme Scale' disponible
Modifiez les paramètres 'Serveurs de grille WebSphere eXtreme Scale', [1=Ajouter, 2=Modifier, 3=Supprimer, 4=Suivant, 5=Quitter]

 (la sélection par défaut est : 4) : 1
Serveurs de grille WebSphere eXtreme Scale (la sélection par défaut est ) : rhea00b02_c0
Adresse URL de service JMX (la valeur par défaut est : service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer) :
service:jmx:rmi:///jndi/rmi://localhost:15000/objectgrid/MBeanServer

'Paramètres 'Serveurs de grille WebSphere eXtreme Scale' : Serveurs de grille WebSphere eXtreme Scale=ogx
Modifiez les paramètres 'Serveurs de grille WebSphere eXtreme Scale', [1=Ajouter, 2=Modifier, 3=Supprimer, 4=Suivant, 5=Quitter]

 (la sélection par défaut est : 4) : 1
Serveurs de grille WebSphere eXtreme Scale (la sélection par défaut est ) : rhea00b02_c1
Adresse URL de service JMX (la valeur par défaut est : service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer) :
service:jmx:rmi:///jndi/rmi://localhost:15001/objectgrid/MBeanServer

'Paramètres 'Serveurs de grille WebSphere eXtreme Scale' : Serveurs de grille WebSphere eXtreme Scale= rhea00b02_c1
Modifiez les paramètres 'Serveurs de grille WebSphere eXtreme Scale', [1=Ajouter, 2=Modifier, 3=Supprimer, 4=Suivant, 5=Quitter]
(la sélection par défaut est : 4) : 1
Serveurs de grille WebSphere eXtreme Scale (la sélection par défaut est ) : rhea00b02_c2
Adresse URL de service JMX (la valeur par défaut est : service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer) :
service:jmx:rmi:///jndi/rmi://localhost:15002/objectgrid/MBeanServer

'Paramètres 'Serveurs de grille WebSphere eXtreme Scale' : Serveurs de grille WebSphere eXtreme Scale= rhea00b02_c2
Modifiez les paramètres 'Serveurs de grille WebSphere eXtreme Scale', [1=Ajouter, 2=Modifier, 3=Supprimer, 4=Suivant, 5=Quitter]

 (la sélection par défaut est : 4) : 5

Cet agent se connectera-t-il à un TEMS ? [1=OUI, 2=NON] (la sélection par défaut est 1) :
Nom d'hôte TEMS (la sélection par défaut est rhea00b00) :

Protocole de réseau [ip, sna, ip.pipe ou ip.spipe] (la sélection par défaut est ip.pipe) :

     Choisissez maintenant le prochain numéro de protocole parmi l'un des suivants :
     - ip
     - sna
     - ip.spipe
     - 0 pour aucun
Protocole de réseau 2 (la sélection par défaut est 0) :
Numéro de port IP.PIPE (la sélection par défaut est 1918) :
Entrez le nom de KDC_PARTITION (la sélection par défaut est null) :

Configurer la connexion TEMS secondaire ? [1=OUI, 2=NON] (la sélection par défaut est 2) :
Entrez Nom du réseau primaire optionnel ou 0 pour "aucun" (la sélection par défaut est 0) :
Configuration de l'agent terminée...
```

L'exemple précédent crée une instance d'agent appelée “inst1” et met à jour les paramètres de Java Home. Les serveurs de conteneur eXtreme Scale sont configurés, mais le service de catalogue n'est pas configuré.

Remarque : La procédure précédente crée un fichier texte au format suivant dans le répertoire : `<install_ITM>/config/<hôte>_xt_<nom de l'instance>.cfg`.

Exemple : `rhea00b02_xt_inst1.cfg`

Il est recommandé d'éditer ce fichier à l'aide de l'éditeur de texte en clair de votre choix. Voci un exemple de contenu d'un tel fichier :

```
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 }  ]]
```

Voici un exemple illustrant une configuration sur un déploiement WebSphere Application Server :

```
rhea00b02 # ./itmcmd config -A xt
Configuration de l'agent démarrée...
Entrez un nom d'instance (la valeur par défaut est : ) : inst1
Modifier les paramètres "Monitoring Agent for WebSphere eXtreme Scale" ? [ 1=Oui, 2=Non ] (la sélection par défaut est 1) : 1
Modifier les paramètres 'Java' ? [ 1=Oui, 2=Non ] (la sélection par défaut est 1) : 1
Répertoire de base Java (la sélection par défaut est C:\Program Files\IBM\Java50) : /opt/WAS61/java
Niveau de trace Java [ 1=Erreur, 2=Avertissement, 3=Informations, 4=Débogage minimum, 5=Débogage moyen, 6=Débogage maximum,
    7=Tous ] (la sélection par défaut est 1) :
Arguments JVM (la sélection par défaut est ):
Modifier les paramètres 'Connexion' ? [ 1=Oui, 2=Non ] (la sélection par défaut est 1) :
Type de connexion de serveur MBean [ 1=Serveur conforme à JSR-160, 2=WebSphere Application Server version 6.0,
3=WebSphere Application Server version 6.1, 4=WebSphere Application Server version 7.0 ] (la sélection par défaut est 1) : 4
Modifier les paramètres 'WebSphere Application Server version 7.0' ? [ 1=Oui, 2=Non ] (la sélection par défaut est 1) :ID utilisateur WAS
(la sélection par défaut est ) :
Entrez Mot de passe WAS (la sélection par défaut est ) :
Entrez de nouveau : Mot de passe WAS (la sélection par défaut est ) :
Nom d'hôte WAS (la sélection par défaut est localhost) : rhea00b02
Port WAS (la sélection par défaut est 2809) :
Protocole de connecteur WAS [ 1=rmi, 2=soap ] (la sélection par défaut est 1) :
Nom de profil WAS (la sélection par défaut est ) : valeur par défaut
----------------------------------------
Informations de chemin de classe WAS
Chemins de base WAS (la sélection par défaut est C:\Program Files\IBM\WebSphere\AppServer;/opt/IBM/WebSphere/AppServer) : /opt/WAS61
Chemin de classes WAS (la sélection par défaut est runtimes/com.ibm.ws.admin.client_6.1.0.jar;runtimes/com.ibm.ws.ejb.thinclient_7.0.0.jar) :
Répertoires JAR WAS (la sélection par défaut est lib;plugins) :
Modifier les paramètres 'Serveurs de grille WebSphere eXtreme Scale' ? [ 1=Oui, 2=Non ] (la sélection par défaut est 1) :
Aucun paramètre 'Serveurs de grille WebSphere eXtreme Scale' disponible
Modifiez les paramètres 'Serveurs de grille WebSphere eXtreme Scale', [1=Ajouter, 2=Modifier, 3=Supprimer, 4=Suivant, 5=Quitter]

(la sélection par défaut est : 4) : 1
Serveurs de grille WebSphere eXtreme Scale (la sélection par défaut est ) : rhea00b02
Adresse URL de service JMX (la valeur par défaut est : service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer) :

Paramètres 'Serveurs de grille WebSphere eXtreme Scale' : Serveurs de grille WebSphere eXtreme Scale=rhea00b02
Modifiez les paramètres 'Serveurs de grille WebSphere eXtreme Scale', [1=Ajouter, 2=Modifier, 3=Supprimer, 4=Suivant, 5=Quitter]
(la sélection par défaut est : 4) : 5
Modifier les paramètres 'Service de catalogue WebSphere eXtreme Scale' ? [ 1=Oui, 2=Non ] (la sélection par défaut est 1) : 2
Cet agent se connectera-t-il à un TEMS ? [1=OUI, 2=NON] (la sélection par défaut est 1) :
Nom d'hôte TEMS (la sélection par défaut est rhea00b02) :

Protocole de réseau [ip, sna, ip.pipe ou ip.spipe] (la sélection par défaut est ip.pipe) :

     Choisissez maintenant le prochain numéro de protocole parmi l'un des suivants :
     - ip
     - sna
     - ip.spipe
     - 0 pour aucun
Protocole de réseau 2 (la sélection par défaut est 0) :
Numéro de port IP.PIPE (la sélection par défaut est 1918) :
Entrez le nom de KDC_PARTITION (la sélection par défaut est null) :

Configurer la connexion TEMS secondaire ? [1=OUI, 2=NON] (la sélection par défaut est 2) :
Entrez Nom du réseau primaire optionnel ou 0 pour "aucun" (la sélection par défaut est 0) :
Configuration de l'agent terminée...
rhea00b02 #
```

Pour les déploiements WebSphere Application Server, vous n'avez pas besoin de créer plusieurs sous-noeuds. L'agent eXtreme Scale se connecte à l'agent de noeud pour collecter toutes les informations des serveurs d'applications dont il est responsable.

`SECTION=CAT` signifie une ligne de service de catalogue, tandis que `SECTION=OGS` signifie une ligne de configuration de serveur eXtreme Scale.

5. Configurez le port JMX pour tous les serveurs conteneurs eXtreme Scale.

   Si des serveurs conteneurs eXtreme Scale sont démarrés, sans l'argument **-JMXServicePort**, un serveur MBean reçoit un port dynamique. L'agent doit savoir à l'avance avec quel port JMX communiquer. L'agent ne fonctionne pas avec des ports dynamiques.

   Au démarrage des serveurs, vous devez spécifier l'argument `-JMXServicePort <numéro_port>` lorsque vous démarrez le serveur eXtreme Scale à l'aide de la commande `startOgServer.sh | .bat`. L'exécution de cette commande garantit que le serveur JMX du processus écoute sur un port statique prédéfini.

Pour les exemples précédents d'une installation UNIX, deux serveurs eXtreme Scale doivent être démarrés avec des ports définis :

a. "-JMXServicePort" "15000" (pour rhea00b02_c0)

b. "-JMXServicePort" "15001" (pour rhea00b02_c1)

a. Démarrez l'agent eXtreme Scale.

   En supposant que l'instance `inst1` ait été créée, comme dans l'exemple précédent, exécutez les commandes ci-après.

   1) `cd <install_ITM>/bin`

   2) `itmcmd agent –o inst1 start xt`

b. Arrêtez l'agent eXtreme Scale.

   En supposant que l'instance “inst1” correspond à l'instance créée, comme dans l'exemple précédent, exécutez les commandes ci-après.

   1) `cd <install_ITM>/bin`

   2) `itmcmd agent –o inst1 stop xt`

6. Activez les statistiques pour tous les serveurs conteneurs eXtreme Scale.

   Pour enregistrer les statistiques, l'agent utilise les beans gérés de statistiques eXtreme Scale. La spécification des statistiques eXtreme Scale doit être activée à l'aide de l'une des méthodes suivantes :

   - en configurant les propriétés des serveurs pour activer toutes les statistiques au démarrage de la totalité des serveurs : all=enabled
   - à l'aide de l'utilitaire d'exemple xsadmin pour activer les statistiques pour tous les conteneurs actifs : paramètres -setstatsspec all=enabled

Résultats

Une fois que tous les serveurs sont configurés et démarrés, les données des beans gérés sont affichées sur la console d'IBM Tivoli Portal. Les espaces de travail prédéfinis montrent les graphiques et mesures de données au niveau de chaque noeud.

Les espaces de travail suivants sont définis : **noeud eXtreme Scaleserveurs de grilles** pour tous les noeuds surveillés.

- vue Transactions eXtreme Scale
- vue Fragment primaire eXtreme Scale
- vue Mémoire eXtreme Scale
- vue ObjectMap eXtreme Scale

Vous pouvez également configurer vos propres espaces de travail. Pour plus d'informations, reportez-vous aux informations sur la personnalisation des espaces de travail, dans le centre de documentation d'IBM Tivoli Monitoring.

Surveillance des applications eXtreme Scale à l'aide de CA Wily Introscope

CA Wily Introscope est un produit de gestion tiers qui permet de détecter et de diagnostiquer les problèmes de performances dans les environnements d'application d'entreprise. eXtreme Scale inclut des détails sur la configuration de CA Wily Introscope pour introspecter certaines portions de l'environnement d'exécution de eXtreme Scale afin d'afficher et de valider rapidement les applications eXtreme Scale. CA Wily Introscope fonctionne de manière efficace pour les déploiements autonomes et WebSphere Application Server.

Présentation

Pour surveiller les applications eXtreme Scale avec CA Wily Introscope, vous devez placer des paramètres dans les fichiers PBD (ProbeBuilderDirective) qui vous permettent d'accéder aux informations de surveillance de eXtreme Scale.

Avertissement : Les points d'instrumentation d'Introscope peuvent changer avec chaque corriger ou version. Lorsque vous installez un nouveau groupe de correctifs ou une nouvelle version, recherchez dans la documentation les modifications apportées aux points d'instrumentation.

Vous pouvez configurer des fichiers PBD (ProbeBuilderDirective) de CA Wily Introscope pour surveiller vos applications eXtreme Scale. CA Wily Introscope est un produit de gestion des applications à l'aide duquel vous pouvez détecter, prioritiser et diagnostiquer de manière proactive les problèmes de performances dans vos environnements d'application Web, composite et complexe.

Paramètres des fichiers PBD pour la surveillance du service de catalogue

Vous pouvez utiliser un ou plusieurs des paramètres ci-après dans votre fichier PBD pour surveiller le service de catalogue.

```
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}"
```

Classes de surveillance du service de catalogue

HAControllerImpl
La classe HAControllerImpl gère le cycle de vie du groupe central et les événements de retour d'informations. Vous pouvez surveiller cette classe pour déterminer les modifications et la structure du groupe central.

ServerAgent
La classe ServerAgent est chargée de communiquer les événements du groupe central avec le service de catalogue. Vous pouvez surveiller les divers appels de signal de présence pour identifier les événements principaux.

PlacementServiceImpl
La classe PlacementServiceImpl coordonne les conteneurs. Vous pouvez utiliser les méthodes de cette classe pour surveiller les événements de jointure et de positionnement.

BalanceGridEventListener
La classe BalanceGridEventListener contrôle la position de leader du catalogue. Vous pouvez surveiller cette classe pour déterminer quel service de catalogue sert actuellement de leader.

Paramètres des fichiers PBD pour la surveillance des conteneurs

Vous pouvez utiliser un ou plusieurs des paramètres ci-après dans votre fichier PBD pour surveiller les conteneurs.

```
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}"
```

Classes de surveillance des conteneurs

ShardImpl
La classe ShardImpl contient la méthode processMessage. La méthode processMessage est celle des demandes client. Avec cette méthode, vous pouvez obtenir les temps de réponse côté serveur et le nombre de demandes. En observant les résultats sur tous les serveurs et en surveillant l'utilisation des segments de mémoire, vous pouvez déterminer si la grille est équilibrée.

CheckpointIterator
La classe CheckpointIterator contient l'appel de méthode activateListener qui place les fragments primaires en mode homologue. Lorsque les fragments primaires sont placés en mode homologue, le fragment réplique est au même niveau que le fragment primaire une fois la méthode exécutée. Lorsqu'une réplique est régénérée à partir d'un fragment primaire complet, cette opération peut durer un certain temps. Le système n'ayant pas intégralement récupéré tant que cette opération n'est pas terminée, vous pouvez utiliser cette classe pour surveiller la progression de l'opération.

CommittedLogSequenceListenerProxy
La classe CommittedLogSequenceListenerProxy contient deux méthodes intéressantes. La méthode applyCommitted est exécutée pour chaque transaction et la méthode sendApplyCommitted est exécutée lorsque le fragment réplique extrait des informations. Le ratio de fréquence d'exécution de ces deux méthodes peut vous indiquer dans quelle mesure le fragment réplique est capable de suivre le fragment primaire.

Paramètres des fichiers PBD pour la surveillance des clients

Vous pouvez utiliser un ou plusieurs des paramètres ci-après dans votre fichier PBD pour surveiller les clients.

```
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}"
```

Classes de surveillance des clients

ORBClientCoreMessageHandler
La classe ORBClientCoreMessageHandler est chargée d'envoyer les demandes d'application aux conteneurs. Vous pouvez surveiller la méthode sendMessage pour le temps de réponse des clients et le nombre de demandes.

ClusterStore
La classe ClusterStore contient les informations de routage côté client.

BaseMap
La classe BaseMap contient la méthode evictMapEntries qui est appelée lorsque l'expulseur souhaite supprimer des entrées de la mappe.

SelectionServiceImpl
La classe SelectionServiceImpl effectue les décisions de routage. Si le client décide de procéder à un basculement, vous pouvez utiliser cette classe pour afficher les actions réalisées à partir de ces décisions.

ObjectGridImpl
La classe ObjectGridImpl contient la méthode getSession que vous pouvez surveiller pour afficher le nombre de demandes pour cette méthode.

Surveillance d'eXtreme Scale à l'aide de Hyperic HQ

Hyperic HQ est une solution de surveillance tiers disponible gratuitement comme solution à code source ouvert ou produit d'entreprise. WebSphere eXtreme Scale inclut un plug-in qui permet aux agents Hyperic HQ de reconnaître les serveurs conteneurs eXtreme Scale et de fournir et regrouper des statistiques à l'aide de beans de gestion eXtreme Scale. Vous pouvez utiliser Hyperic HQ pour surveiller les déploiements eXtreme Scale autonomes.

Avant de commencer

- Ce jeu d'instructions concerne Hyperic Version 4.0. Si vous possédez une version plus récente de Hyperic, reportez-vous à la documentation de Hyperic pour plus d'informations, telles que les noms de chemin et la méthode de démarrage des agents et des serveurs.
- Téléchargez les installations des agents et serveurs Hyperic. Une installation de serveur doit être en cours d'exécution. Pour détecter tous les serveurs eXtreme Scale, un agent Hyperic doit être en cours d'exécution sur chaque machine sur laquelle un serveur eXtreme Scale est en cours d'exécution. Pour les informations de téléchargement et le support de documentation, voir le site Web Hyperic.
- Vous devez avoir accès aux fichiers `objectgrid-plugin.xml` et `hqplugin.jar`. Ces fichiers se trouvent dans le répertoire `objectgridRoot/hyperic/etc`.

Pourquoi et quand exécuter cette tâche

En intégrant eXtreme Scale au logiciel de surveillance Hyperic HQ, vous pouvez surveiller et afficher graphiquement les mesures sur les performances de votre environnement. Vous configurez cette intégration en utilisant une implémentation de plug-in sur chaque agent.

Procédure

1. Démarrez vos serveurs eXtreme Scale. Le plug-in Hyperic recherche les processus locaux à connecter aux machines virtuelles Java qui exécutent eXtreme Scale. Pour se connecter correctement aux machines virtuelles Java, chaque serveur doit être démarré avec l'option **-jmxServicePort**. Pour plus d'informations sur le démarrage des serveurs à l'aide de l'option **-jmxServicePort**, voir «Script startOgServer», à la page 336.
2. Placez le fichier `extremescale-plugin.xml` et le fichier `wxshyperic.jar` dans les répertoires de plug-in appropriés du serveur et des agents, dans votre configuration Hyperic. Pour être intégrées à Hyperic, les installations des agents et du serveur doivent avoir accès au plug-in et aux fichiers JAR (archive Java). Le serveur peut permuter dynamiquement les configurations, mais vous devez effectuer l'intégration avant de démarrer l'un des agents.
   a. Placez le fichier `extremescale-plugin.xml` dans le répertoire `plugin` du serveur, qui se trouve à l'emplacement suivant :

      `base_hyperic/base_serveur/hq-engine/server/default/deploy/hq.ear/hq-plugins`
   b. Placez le fichier `extremescale-plugin.xml` dans le répertoire `plugin` de l'agent, qui se trouve à l'emplacement suivant :

      `base_agent/bundles/gent-4.0.2-939/pdk/plugins`
   c. Placez le fichier `wshyperic.jar` dans le répertoire `lib` de l'agent, qui se trouve à l'emplacement suivant :

      `base_agent/bundles/gent-4.0.2-939/pdk/lib`
3. Configurez l'agent. Le fichier `agent.properties` sert de point de configuration pour l'environnement d'exécution de l'agent. Cette propriété se trouve dans le répertoire *rép_base_agent*`/conf`. Les clés suivantes sont facultatives, mais importantes pour le plug-in eXtreme Scale :
   - `autoinventory.defaultScan.interval.millis=<`*durée_en_millisecondes*`>`

     Définit l'intervalle en millisecondes entre les reconnaissances d'Agent.
   - `log4j.logger.org.hyperic.hq.plugin.extremescale.XSServerDetector=DEBUG`

     : Active les instructions de débogage prolixes à partir du plug-in eXtreme Scale.
   - `username=<`*nomutilisateur*`>` : Définit le nom d'utilisateur JMX (Java Management Extensions) si la sécurité est activée.
   - `password=<`*motdepasse*`>` : Définit le mot de passe JMX si la sécurité est activée.
   - `sslEnabled=<`*true*`|`*false*`>` : Indique au plug-in s'il doit utiliser SSL (Secure Sockets Layer). La valeur est `false` par défaut.
   - `trustPath=<`*chemin*`>` : Définit le chemin sécurisé de la connexion SSL.
   - `trustType=<`*type*`>` : Définit le type sécurisé de la connexion SSL.
   - `trustPass=<`*motdepasse*`>` : Définit le mot de passe sécurisé de la connexion SSL.

4. Démarrez la reconnaissance des agents. Les agents Hyperic envoient des informations de reconnaissance et des mesures au serveur. Utilisez le serveur pour personnaliser les vues de données et regrouper les objets d'inventaire logiques afin de générer des informations utiles. Une fois que le serveur est disponible, vous devez exécuter le script de lancement ou démarrer le service Windows de l'agent :

   - Linux `base_agent/bin/hq-agent.sh start`
   - Windows Démarrez l'agent avec le service Windows.

   Une fois que vous avez démarré les agents, les serveurs sont détectés et les groupes sont configurés. Vous pouvez vous connecter à la console du serveur et choisir les ressources à ajouter à la base de données d'inventaire du serveur. La console du serveur se trouve à l'URL suivante par défaut : `http://<nom_hôte_serveur>:7080/`
5. Les statistiques doivent être activées pour que Hyperic puisse collecter des données statistiques.

   Utilisez l'action de contrôle **SetStatsSpec** sur la console Hyperic pour eXtreme Scale. Allez à la ressource, puis utilisez la liste déroulante **Action de contrôle** sous l'onglet **Contrôle** afin de spécifier un paramètre `SetStatsSpec` avec `ALL=enabled` dans la zone de texte **Arguments du contrôle**.

   Les serveurs de catalogues ne sont pas détectés par le filtre défini sur la console Hyperic. Voir les informations concernant la propriété **statsSpec** dans «Fichier de propriétés du serveur», à la page 185, qui activent les statistiques au démarrage des conteneurs. Diverses options d'activation des statistiques sont décrites dans «Surveillance à l'aide de beans gérés (MBeans)», à la page 402 et dans «Surveillance à l'aide de l'exemple d'utilitaire xsAdmin», à la page 387.
6. Surveillez les serveurs à l'aide de la console Hyperic. Une fois que les serveurs ont été ajoutés au modèle d'inventaire, leurs services ne sont plus requis.
   - **Vue Tableau de bord** : Lorsque vous avez affiché les événements de détection des ressources, vous vous êtes connecté à la vue du tableau de bord principal. Il s'agit d'une vue générique qui sert de centre de messagerie que vous pouvez personnaliser. Vous pouvez exporter des graphiques ou des objets d'inventaire dans ce tableau de bord principal.
   - **Vue Ressources** : Vous pouvez interroger et afficher l'intégralité du modèle d'inventaire à partir de cette page. Une fois que les services ont été ajoutés, chaque serveur eXtreme Scale est correctement libellé et répertorié sous la section des serveurs. Vous pouvez cliquer sur chacun des serveurs pour consulter les mesures de base.
7. Affichez l'intégralité de l'inventaire du serveur dans la page d'affichage des ressources. Dans cette page, vous pouvez sélectionner plusieurs serveurs ObjectGrid et les regrouper. Une fois que vous avez regroupé un ensemble de ressources, leurs mesures communes peuvent être représentées graphiquement pour montrer les superpositions et les différences entre les membres du groupe. Pour afficher une superposition, sélectionnez les mesures dans l'écran de votre groupe de serveurs. La mesure est affichée dans la zone de représentation graphique. Pour afficher une superposition pour tous les membres du groupe, cliquez sur le nom de mesure souligné. Vous pouvez exporter les graphiques, vues de noeud et superpositions comparatives de votre choix dans le tableau de bord principal, à l'aide du menu **Outils**.

Chapitre 10. Optimisation et performances

Liste de contrôle opérationnelle

Utilisez la liste de contrôle opérationnelle afin de préparer votre environnement pour le déploiement de WebSphere eXtreme Scale.

Tableau 27. Liste de contrôle opérationnelle

Elément de la liste de contrôle	Pour plus d'informations

Si vous utilisez AIX, optimisez les paramètres suivants du système d'exploitation :
 	
TCP_KEEPINTVL
Le paramètre TCP_KEEPINTVL fait partie d'un protocole de maintien de connexion du socket qui permet la détection des indisponibilités du réseau. La propriété spécifie l'intervalle entre les paquets envoyés pour valider la connexion. Si vous utilisez WebSphere eXtreme Scale, spécifiez la valeur `10`. Pour vérifier le paramètre actuel, exécutez la commande suivante :
`# no –o tcp_keepintvl`
Pour modifier le paramètre actuel, exécutez la commande suivante :
`# no –o tcp_keepintvl=10`
Le paramètre TCP_KEEPINTVL est exprimé en demi secondes.
 	
TCP_KEEPINIT
Le paramètre TCP_KEEPINIT fait partie d'un protocole de maintien de connexion du socket qui permet la détection des indisponibilités du réseau. La propriété spécifie le délai d'attente initial de la connexion TCP. Si vous utilisez WebSphere eXtreme Scale, spécifiez la valeur `40`. Pour vérifier le paramètre actuel, exécutez les commandes suivantes :
`# no –o tcp_keepinit`
Pour modifier le paramètre actuel, exécutez la commande suivante :
`# no –o tcp_keepinit=40`
Le paramètre TCP_KEEPINIT est exprimé en demi secondes.
	• Pour des informations sur l'optimisation d'AIX, voir la rubrique Optimisation des systèmes AIX.

Mettez à jour le fichier `orb.properties` pour modifier le comportement de transport de la grille. Le fichier `orb.properties` se trouve dans le répertoire `java/jre/lib`.	«Fichier de propriétés de l'ORB», à la page 195

Tableau 27. Liste de contrôle opérationnelle (suite)

Elément de la liste de contrôle	Pour plus d'informations
Utilisez les paramètres du script startOgServer. En particulier, utilisez les paramètres suivants : • Définissez les paramètres de segment de mémoire avec l'option **-jvmArgs**. • Définissez les propriétés et le chemin d'accès aux classes avec l'option **-jvmArgs**. • Définissez les paramètres **-jvmArgs** pour configurer la surveillance de l'agent. **Paramètres de port** WebSphere eXtreme Scale doit ouvrir des ports pour les communications pour certains transports. Ces ports sont tous définis de manière dynamique. Toutefois, si un pare-feu est utilisé entre les conteneurs, vous devez spécifier les ports. Utilisez les informations suivantes sur les ports : **Port du programme d'écoute** Vous pouvez utiliser l'argument **-listenerPort** pour spécifier le port utilisé pour les communications entre les processus. **Port du groupe central** Vous pouvez utiliser l'argument **-haManagerPort** pour spécifier le port utilisé pour la détection des incidents. Cet argument correspond à peerPort. Notez que les groupes centraux n'ayant pas besoin de communiquer entre les zones, il n'est pas nécessaire de définir ce port si le pare-feu est ouvert pour tous les membres d'une même zone. **Port du service JMX** Vous pouvez utiliser l'argument **-JMXServicePort** pour spécifier le port à utiliser par le service JMX. **Port SSL** Si vous transmettez `-Dcom.ibm.CSI.SSLPort=1234` comme argument **-jvmArgs**, le port SSL prend la valeur `1234`. Le port SSL est l'homologue de port sécurisé du port du programme d'écoute. **Port du client** Utilisé uniquement dans le service de catalogue. Vous pouvez spécifier cette valeur avec l'argument **-catalogServiceEndPoints**. La valeur de ce paramètre est au format suivant : `nomServeur:nomHôte:portClient:portHomologue`	«Script startOgServer», à la page 336
Vérifiez que les paramètres de sécurité sont configurés correctement : • Transport (SSL) • Application (Authentification et autorisation) Pour vérifier vos paramètres de sécurité, vous pouvez essayer d'utiliser un client malveillant pour vous connecter à votre configuration. Par exemple, si le paramètre SSL requis est configuré, un client possédant un paramètre TCP_IP avec ce dernier ou un client possédant un fichier de clés certifiées incorrect ne doit pas pouvoir se connecter au serveur. Si l'authentification est requise, un client sans données d'identification, telles qu'un ID utilisateur et un mot de passe ne doit pas pouvoir se connecter au serveur. Si l'autorisation est appliquée, un client sans autorisation d'accès ne doit pas être autorisé à accéder aux ressources du serveur.	«Intégration de la sécurité à des fournisseurs externes», à la page 370
Choisissez comment vous allez surveiller votre environnement. • xsAdmin – Les ports JMX des serveurs de catalogues doivent être visibles par l'outil XSAdmin. Les ports de conteneur doivent également être accessibles pour certaines commandes qui collectent des informations des conteneurs. • Vous pouvez choisir entre les outils de surveillance de fournisseur suivants : – Tivoli Enterprise Monitoring Agent – CA Wily Introscope – Hyperic HQ	• «Surveillance à l'aide de l'exemple d'utilitaire xsAdmin», à la page 387 • «Sécurité JMX (Java Management Extensions)», à la page 368 • «Surveillance à l'aide d'IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale», à la page 410 • «Surveillance d'eXtreme Scale à l'aide de Hyperic HQ», à la page 420 • «Surveillance des applications eXtreme Scale à l'aide de CA Wily Introscope», à la page 416

Optimisation des systèmes d'exploitation et des réseaux

En modifiant les paramètres de connexion, il est possible de réduire les temps d'attente TCP (Transmission Control Protocol) et la modification des tampons TCP permet d'améliorer les débits de transmission.

Systèmes d'exploitation

De tous les systèmes d'exploitation, Windows est celui qui a le moins besoin d'être optimisé, au contraire de Solaris, qui nécessite un maximum d'optimisation. Les informations suivantes, qui concernent chacun des systèmes spécifiés, sont susceptibles d'améliorer les performances de WebSphere eXtreme Scale. Vous devez procéder à l'optimisation en fonction de la charge de votre réseau et de vos applications.

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
```

Planification des ports réseau

WebSphere eXtreme Scale est un cache réparti devant ouvrir des ports pour communiquer avec l'ORB (Object Request Broker) et la pile TCP (Transmission Control Protocol) sur les machines virtuelles Java et les autres machines. Vous devez planifier et contrôler les ports, particulièrement dans un environnement de pare-feu, par exemple lorsque vous utilisez un service de catalogue et des conteneurs sur plusieurs ports.

Domaine de services de catalogue

Un domaine de services de catalogue nécessite que soient définis les ports suivants :

peerPort
Spécifie le port qui permet au gestionnaire de haute disponibilité (HA) de communiquer entre serveurs de catalogue homologues dans une pile TCP.

clientPort
Spécifie le port qui permet aux serveurs de catalogue d'accéder aux données des services de catalogue.

JMXServicePort
Spécifie le port que doit utiliser le service JMX (Java Management Extensions).

listenerPort
Définit le port d'écoute ORB qui permet aux conteneurs et aux clients de communiquer avec le service de catalogue via l'ORB.

La manière dont vous définissez ces ports dépend du mode que vous utilisez : autonome ou démarrage des serveurs de catalogue eXtreme Scale dans un environnement WebSphere Application Server :

- **En mode autonome :**

  Utilisez la commande startOgServer pour définir les ports indiqués plus haut avec l'option autonome, comme dans l'exemple suivant :

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

  Voir «Démarrage du service de catalogue dans un environnement autonome», à la page 331 pour en savoir plus sur le démarrage du service de catalogue en mode autonome.
- **Pour un environnement WebSphere Application Server :**

  Vous pouvez définir un domaine de services de catalogue dans la console d'administration. Pour plus d'informations, voir «Création de domaines de service de catalogue dans WebSphere Application Server», à la page 346.

Serveurs conteneurs

Les serveurs conteneurs WebSphere eXtreme Scale requièrent également que plusieurs ports soient en fonctionnement. Par défaut, le serveur conteneur eXtreme Scale génère automatiquement son port de gestionnaire HA et son port d'écoute ORB avec des ports dynamiques. Dans un environnement de pare-feu, il est avantageux pour vous de planifier et de contrôler les ports. Certaines options vous permettent donc de démarrer les serveurs conteneurs eXtreme Scale avec le port du gestionnaire HA et le port d'écoute ORB à l'aide d'une option de la commande startOgServer, comme dans l'exemple ci-dessous :

```
-HaManagerPort <port_homologue>
-listenerPort <portORB>
```

Une planification optimale des ports est un avantage, mais il est difficile de les planifier et de les gérer lorsque des centaines de machines virtuelles Java sont démarrées dans une machine. Un conflit de port entraîne un échec de démarrage du serveur.

Lorsque la sécurité est activée, un port SSL (Secure Socket Layer) doit être ajouté aux ports répertoriés précédemment. L'utilisation de `Dcom.ibm.CSI.SSLPort=<sslPort>` comme argument **-jvmArgs** donne au port SSL la valeur de `<sslPort>`. Pour obtenir de l'aide sur la planification des ports, consultez les paramètres de sécurité dans eXtreme Scale.

Propriétés ORB et paramétrage du descripteur de fichiers

L'optimisation inclut également les propriétés Object Request Broker (ORB) et le paramétrage du descripteur de fichiers.

Propriétés ORB

L'ORB sert à WebSphere eXtreme Scale pour communiquer dans l'ensemble d'une pile TCP. L'indispensable fichier `orb.properties` se trouve dans le répertoire `java/jre/lib`. Dans le cas de lourde charge d'objets de grande taille, activez la fragmentation ORB en spécifiant le paramètre suivant :

```
com.ibm.CORBA.FragmentSize=<taille appropriée>
```

Empêchez la croissance du pool d'unités d'exécution en spécifiant le paramètre suivant :

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

En spécifiant le paramètre suivant, fixez des délais d'expiration appropriés qui vous éviteront d'avoir un trop grand nombre d'unités d'exécution dans des situations anormales :

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

Descripteur de fichiers

Les systèmes UNIX et Linux imposent une limite au nombre de fichiers ouverts autorisé par processus. C'est le système d'exploitation qui spécifie le nombre permis de fichiers ouverts. Si cette valeur est trop basse, une erreur d'allocation de mémoire se produira sur AIX avec consignation d'une erreur de fichiers ouverts en trop grand nombre.

Dans la fenêtre de terminal UNIX, augmentez cette valeur au-dessus de la valeur par défaut du système. Dans le cas de grosses machines SMP avec des clones, fixez une valeur illimitée.

Pour les configurations AIX, fixez une valeur de `-1` (illimitée) à l'aide de la commande `ulimit -n -1`.

Pour les configurations Solaris, fixez une valeur de `16384` (illimitée) à l'aide de la commande `ulimit -n 16384`.

Pour connaître la valeur actuelle, utilisez la commande `ulimit –a`.

NIO avec ORB

A l'heure actuelle, il est possible de fonctionner avec NIO (ChannelFramework) dans des scénarios WebSphere eXtreme Scale autonome. Pour pouvoir fonctionner avec NIO (Non-blocking I/O) dans l'ORB d'IBM, vous devez donner au mode de transport de l'ORB d'IBM la valeur ChannelFramework. En effet, par défaut cet ORB fonctionne en mode Pluggable. WebSphere eXtreme Scale fournit une propriété de serveur permettant de définir le mode de transport comme ChannelFramework

Important :

Les versions actuelles de WebSphere Application Server ne prennent en charge que le mode Pluggable. Lorsque WebSphere eXtreme Scale s'exécute intégré à WebSphere Application Server, il doit se conformer au mode Pluggable. WebSphere eXtreme Scale utilisant également le SSL/Transport Security de WebSphere Application Server, lui aussi ne prend en charge que le mode Pluggable.

Quand utiliser NIO

C'est ce que nous allons exposer succinctement.

Activation de NIO ou de ChannelFramework sur l'ORB

WebSphere eXtreme Scale fournit une propriété de serveur permettant de définir le mode de transport comme étant ChannelFramework afin d'activer NIO (les entrées-sorties non bloquantes) sur l'ORB.

Avant de commencer

Trouvez le fichier de propriétés du serveur ou créez-en un. Vous pouvez activer ChannelFramework sur le service de catalogue et sur les serveurs conteneurs. pour plus de détails, voir Fichier de propriétés de serveur.

Pourquoi et quand exécuter cette tâche

A l'heure actuelle, il est possible de fonctionner avec NIO ou ChannelFramework) dans des scénarios WebSphere eXtreme Scale autonome. Pour pouvoir fonctionner avec NIO dans l'ORB d'IBM, vous devez définir ChannelFramework comme mode de transport de cet ORB. En effet, par défaut cet ORB fonctionne en mode Pluggable. WebSphere eXtreme Scale fournit une propriété de serveur permettant de définir le mode de transport comme ChannelFramework.

Important :

Les versions actuelles de WebSphere Application Server ne prennent en charge que le mode Pluggable. Lorsque WebSphere eXtreme Scale s'exécute intégré à WebSphere Application Server, il doit se conformer au mode Pluggable. WebSphere eXtreme Scale utilisant également le SSL/Transport Security de WebSphere Application Server, lui aussi ne prend en charge que le mode Pluggable.

Procédure

1. Ajoutez la propriété enableChannelFramework=true au fichier de propriétés de votre serveur.
2. Assurez-vous que ce fichier de propriétés ne contredit pas le fichier des propriétés de l'ORB.

Si le fichier de propriétés du serveur active le mode de transport ChannelFramework, mais que ce dernier est défini comme Pluggable dans le fichier orb.properties, le paramétrage du serveur ne prendra pas le pas sur celui de l'ORB. Un message vous avertira dans le journal qu'il existe deux paramétrages. Pour permettre à la propriété enableChannelFramework=true d'entrer en vigueur, ajustez les propriétés qui indiquent que le mode de transport est Pluggable : remplacez com.ibm.CORBA.TransportMode=Pluggable par com.ibm.CORBA.TransportMode=ChannelFramework ou supprimez carrément la propriété.

3. Fournissez le fichier de propriétés du serveur modifié au démarrage du service de catalogue ou du serveur conteneur. Pour plus de détails sur l'utilisation des fichiers de propriétés de serveur pour démarrer un serveur, voir Fichier de propriétés de serveur.

Résultats

Lorsqu'un service de catalogue ou un serveur conteneur utilise le mode de transport channelFramework, il écrit le message suivant dans le journal.

```
CWOBJ0052I: La propriété IBM ORB TransportMode a été associée à ChannelFramework.
```

Si vous voyez dans le journal le message qui suit, examinez les propriétés de votre ORB, comme expliqué précédemment.

```
CWOBJ0055W: La propriété IBM ORB TransportMode a été associée à ChannelFramework
dans le fichiers des propriétés du serveur mais le fichier orb.properties
existant contient déjà un paramètre TransportMode défini.
La propriété TransportMode ne sera pas remplacée.
```

Notez que lorsque vous activez ChannelFramework, la valeur maximum de ServerSocketQueueDepth est de 512. Si le paramètre ServerSocketQueueDepth d'orb.properties est supérieur à 512, le serveur ramènera automatiquement ServerSocketQueueDepth à 512 et vous en avisera en écrivant un message d'information dans le journal. Aucune action de votre part n'est nécessaire.

```
CWOBJ0053I: La propriété IBM ORB ServerSocketQueueDepth a été associée à 512

pour s'exécuter correctement avec la propriété ChannelFramework TransportMode.
```

Optimisation des machines virtuelles Java pour WebSphere eXtreme Scale

L'optimisation des machines virtuelles Java (JVM) peut contribuer à améliorer votre déploiement de WebSphere eXtreme Scale de manière considérable.

Dans la plupart des cas, un faible nombre de paramètres JVM ou des paramètres JVM peu spécifiques sont requis par WebSphere eXtreme Scale. Si une quantité importante d'objets sont stockés dans WebSphere eXtreme Scale, ajustez la taille de pile à un niveau approprié pour que la mémoire ne devienne pas insuffisante.

Plateformes

Le test de performances a lieu principalement sur les sous-systèmes de stockage AIX (32 voies), Linux (4 voies) et Windows (8 voie). Avec les systèmes AIX haut de gamme, les scénarios à multiples unités d'exécution peuvent être appliqués et les points de conflit peuvent être identifiés et résolus.

Exigences de la fonction ORB (Object Request Broker)

Le kit de développement de logiciels IBM comprend une implémentation ORB IBM qui a été testée avec WebSphere Application Server et WebSphere eXtreme Scale. Pour faciliter le processus de prise en charge, utilisez une machine virtuelle Java IBM. Les autres implémentations de machines virtuelles Java utilisent une autre fonction ORB. La fonction ORB IBM est uniquement fournie prête à l'emploi avec les machines virtuelles Java IBM. WebSphere eXtreme Scale requiert une fonction ORB opérationnelle. Vous pouvez utiliser WebSphere eXtreme Scale avec des fonctions ORB tierces mais si un problème est identifié dans l'ORB, vous devez contacter le fournisseur ORB pour obtenir de l'assistance. L'implémentation ORB IBM est compatible avec des machines virtuellesJava tierces et peut être remplacée si nécessaire.

Récupération de place

WebSphere eXtreme Scale crée des objets temporaires associés à chaque transaction tels que la demande et la réponse ainsi la séquence de journal. Ces objets affectant l'efficacité de la récupération de place, l'optimisation de la récupération de place est cruciale.

Pour la machine virtuelle IBM pour Java, utilisez le collecteur optavgpause pour les cas de fréquence d'actualisation élevée (100 % des transactions modifient les entrées). Le collecteur gencon est plus performant que le collecteur optavgpause pour les scénarios où les données sont mises à jour de façon relativement peu fréquente (fréquence de 10 % ou moins). Expérimentez les deux collecteurs pour savoir lequel est le mieux adapté à vos besoins. Si vous constatez un problème de performance, exécutez le collecteur avec la fonction de récupération de place détaillée pour vérifier la durée (pourcentage) d'exécution de la récupération de place. Des cas ont été relevés où 80 % de l'exécution sont consacrés à la récupération de place jusqu'à ce que l'optimisation corrige le problème.

Pour plus d'informations sur la configuration de la récupération de place, consultez Optimisation de la machine virtuelle IBM pour Java.

Performance de la JVM

WebSphere eXtreme Scale peut être exécuté sur différentes versions de Java 2 Platform, Standard Edition (J2SE). WebSphere eXtreme Scale version 6.1 prend en charge J2SE version 1.4.2 et ultérieures. Pour optimiser la productivité et les performances des développeurs, utilisez J2SE 5 ou version ultérieure pour tirer parti des annotations et de la récupération de place améliorée. WebSphere eXtreme Scale tourne sur les machines virtuelles Java 32 bits ou 64 bits.

WebSphere eXtreme Scale est testé avec un sous-ensemble de machines virtuelles disponibles mais la liste de prise en charge n'est pas exhaustive. Vous pouvez exécuter WebSphere eXtreme Scale sur toute version 1.4.2 ou supérieure mais si un problème est identifié sur la machine virtuelle Java, vous devez contacter le fournisseur JVM pour obtenir de l'assistance. Si possible, utilisez la JVM de l'exécution WebSphere sur toutes les plateformes prises en charge par WebSphere Application Server.

Pour la plupart des scénarios impliquant l'utilisation de WebSphere eXtreme Scale, Java Platform Standard Edition 6 de la JVM est plus performant que Edition 5 ou 1.4. Java 2 Platform, Standard Edition version 1.4 n'est pas performant notamment pour les scénarios qui repose sur le collecteur gencon. Java Platform Standard

Edition 5 est performant mais Java Platform, Standard Edition 6 l'est encore plus.

Segments de mémoire importants

Lorsque l'application doit gérer une large quantité de données pour chaque partition, la récupération de place peut être un facteur déterminant. Un scénario principalement en lecture est performant même avec des segments de mémoire importants (20 Go ou plus) si un collecteur générationnel est utilisé. Toutefois, une fois que le segment de mémoire a été rempli, une interruption proportionnelle à la taille de pile réelle et le nombre de processeurs survient. Cette pause peut être longue sur des serveurs de petite capacité où les segments de mémoire sont de taille importante.

WebSphere eXtreme Scale prend en charge WebSphere Real Time Java. Avec WebSphere Real Time Java, la réponse du traitement des transactions pour WebSphere eXtreme Scale est plus cohérente et prévisible et l'impact de la récupération de place et de la planification des unités d'exécution est considérablement réduite. L'impact est réduit au point où l'écart type du temps de réponse est inférieur à 10 % du langage Java classique.

Pour plus d'informations, voir «Utilisation de WebSphere Real Time», à la page 438.

Nombre d'unités d'exécution

Le nombre d'unités d'exécution dépend de quelques facteurs. Une limite existe pour le nombre d'unités d'exécution pouvant être gérées par un seul fragment. Un fragment est une instance de partition et peut être un fragment primaire ou une réplique. Avec un nombre plus important de fragments pour chaque JVM, plus d'unités d'exécution associées aux fragments supplémentaires fournissent plus de chemins simultanés d'accès aux données. Chaque fragment est aussi concurrent que possible même si l'accès simultané est limité.

Optimisation des machines virtuelles Java

Vous devez prendre en compte plusieurs aspects spécifiques de l'optimisation des machines virtuelles Java (JVM) pour de meilleures performances de WebSphere eXtreme Scale.

La configuration recommandée est d'avoir des segments de 1 à 2 Go avec une machine virtuelle Java pour quatre coeurs. La taille des segments dépend de la nature des objets qui sont stockées sur les serveurs, point que nous aborderons un peu plus loin.

Taille de segment et récupération de place : préconisations

La taille optimale des segments dépend de trois facteurs :

1. le nombre d'objets actifs présents dans le segment
2. le degré de complexité des objets actifs présents dans le segment
3. le nombre de coeurs utilisables par la machine virtuelle Java

Par exemple, une application stockant des tableaux d'octets de 10 Ko pourra exécuter un segment bien plus important qu'une application utilisant des graphes complexes POJO.

Toutes les machines virtuelles Java modernes utilisent aujourd'hui des algorithmes de récupération de place en parallèle ; en d'autres termes, utiliser davantage de coeurs peut réduire les pauses dans la récupération de place. Et la place sera récupérée plus rapidement sur des systèmes 8 coeurs que sur des systèmes quadricoeurs.

Utilisation de la mémoire réelle et taille spécifiée pour les segments

Une machine virtuelle Java avec un segment de 1 Go utilise en fait approximativement 1,3 Go de mémoire réelle. Dans notre lab, nous sommes arrivés à faire tourner des machines virtuelles Java de 1 Go sur un système ayant 16 Go de RAM. Le système a commencé à paginer une fois que les segments ont été pleins jusqu'à 800 Mo ou plus.

Récupération de place

Dans le cas de machines virtuelles Java IBM, utilisez le collecteur avgoptpause pour des scénarios d'actualisations à haut débit (100% des transactions modifient les entrées). Le collecteur gencon fonctionne encore mieux qu'avgoptpause dans le cas de scénarios où les données sont actualisées de manière relativement peu fréquentes (10 % du temps, voire moins). A vous de tester les deux collecteurs pour voir celui qui fonctionne le mieux dans votre cas de figure. Si vous constatez un problème de performances, activez le mode prolixe pour la récupération de place afin de vérifier le pourcentage de temps passé à cette récupération. Il a pu être constaté des scénarios où 80 % du temps était consacré à la récupération de place, jusqu'à ce que l'optimisation règle le problème.

Performances des machines virtuelles Java

WebSphere eXtreme Scale peut s'exécuter sur des versions différentes de Java 2 Platform, Standard Edition (J2SE). ObjectGrid version 6.1 prend en charge J2SE version 1.4.2 et plus récentes. Pour une meilleure productivité de développement et de meilleures performances, utilisez J2SE 5 ou plus récent afin de profiter des annotations et d'une récupération de place améliorée. ObjectGrid tourne sur des machines virtuelles Java 32 ou 64 bits.

Les clients ObjectGrid version 6.0.2 peuvent se connecter à une grille ObjectGrid version 6.1. Utilisez des clients ObjectGrid version 6.1 pour des clients J2SE version 1.4.2 ou mieux. La seule raison d'utiliser un client ObjectGrid version 6.0.2 est sa prise en charge de J2SE version 1.3.

WebSphere eXtreme Scale est testé avec un sous-ensemble des machines virtuelles disponibles, mais la liste des machines prises en charge n'est pas exhaustive. Vous pouvez faire tourner WebSphere eXtreme Scale sur n'importe quelle version 1.4.2 ou supérieure, mais si un problème est rencontré sur la machine virtuelle Java, vous devrez contacter le fournisseur de celle-ci. Dans la mesure du possible, sur les plateformes compatibles WebSphere Application Server, utilisez la machine virtuelle Java provenant de l'environnement d'exécution WebSphere.

La meilleure machine virtuelle Java est la machine Java Platform, Standard Edition 6. Java 2 Platform, Standard Edition, v 1.4 offre de médiocres performances, particulièrement dans les scénarios où le collecteur gencon fait la différence. Les performances de Java Platform Standard Edition 5 sont satisfaisantes mais Java Platform, Standard Edition 6 fonctionne encore mieux.

Optimisation d'orb.properties

Nous recommandons d'utiliser en production le fichier orb.properties qui suit. Dans notre lab, nous avons utilisé ce fichier sur des grilles contenant jusqu'à 1 500 machines virtuelles Java. Le fichier orb.properties se trouve dans le dossier lib du JRE utilisé.

```
# IBM JDK properties for ORB
org.omg.CORBA.ORBClass=com.ibm.CORBA.iiop.ORB
org.omg.CORBA.ORBSingletonClass=com.ibm.rmi.corba.ORBSingleton

# WS Interceptors
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.objectgrid.corba.ObjectGridInitializer

# WS ORB & Plugins properties
com.ibm.CORBA.ForceTunnel=never
com.ibm.CORBA.RequestTimeout=10
com.ibm.CORBA.ConnectTimeout=10

# Needed when lots of JVMs connect to the catalog at the same time
com.ibm.CORBA.ServerSocketQueueDepth=2048

# Clients and the catalog server can have sockets open to all JVMs
com.ibm.CORBA.MaxOpenConnections=1016

# Thread Pool for handling incoming requests, 200 threads here
com.ibm.CORBA.ThreadPool.IsGrowable=false
com.ibm.CORBA.ThreadPool.MaximumSize=200
com.ibm.CORBA.ThreadPool.MinimumSize=200
com.ibm.CORBA.ThreadPool.InactivityTimeout=180000

# No splitting up large requests/responses in to smaller chunks
com.ibm.CORBA.FragmentSize=0
```

Nombre des unités d'exécution

Le nombre d'unités d'exécution dépend de quelques facteurs. Il existe une limite au nombre d'unités d'exécution que peut gérer un seul fragment. Un nombre plus élevé de fragments pour chaque machine virtuelle Java a pour conséquence un nombre plus élevé d'unités d'exécution et donc d'accès simultanés. Chaque fragment supplémentaire apporte davantage de chemins simultanés vers les données. Chaque fragment est aussi simultané que possible, mais, même ici, il existe une limite.

Configuration de la détection des basculements

Le paramètre d'intervalle du signal de présence permet de configurer le laps de temps séparant deux vérifications par le système des serveurs en panne.

Pourquoi et quand exécuter cette tâche

La configuration des basculements varie en fonction du type d'environnement que vous utilisez. Si vous utilisez un environnement autonome, vous pouvez configurer les basculements à l'aide de la ligne de commande. Si vous utilisez un environnement WebSphere Application Server Network Deployment, vous devez les configurer à partir de la console d'administration de WebSphere Application Server Network Deployment.

Procédure

- Configurez les basculements pour les environnements autonomes.

  Vous pouvez configurer les intervalles de signal de présence sur la ligne de commande à l'aide du paramètre **-heartbeat** du fichier script startOgServer.bat | startOgServer.sh. Affectez à ce paramètre l'une des valeurs suivantes :

Tableau 28. Intervalles de signal de présence

Valeur	Action	Description
0	Standard (par défaut)	Les basculements sont généralement détectés dans les 30 secondes.
-1	Elevé	Les basculements sont généralement détectés dans les 5 secondes.
1	Souple	Les basculements sont généralement détectés dans les 180 secondes.

Un intervalle élevé entre les signaux de présence peut être utile si les processus et le réseau sont stables. Si le réseau ou les processus ne sont pas configurés de manière optimale, il peut manquer des signaux de présence, ce qui peut fausser la détection des incidents.

- Configurez les basculements pour les environnements WebSphere Application Server.

  Vous pouvez configurer WebSphere Application Server Network Deployment Version 6.0.2 ou ultérieure pour permettre des basculements très rapides de WebSphere eXtreme Scale. La durée par défaut de pour les incidents matériels est d'environ 200 secondes. Par incident matériel, l'on entend une panne d'ordinateur ou de serveur, une déconnexion de câble réseau ou une erreur du système d'exploitation. Les incidents dus aux pannes de processus ou à des échecs logiciels sont généralement basculés en moins d'une seconde. La détection des incidents logiciels est effectuée lorsque les sockets réseau du processus inactif sont fermés automatiquement par le système d'exploitation du serveur qui héberge le processus.

  Configuration des signaux de présence du groupe central

  Si WebSphere eXtreme Scale est exécuté dans un processus WebSphere Application Server, il hérite des caractéristiques de reprise en ligne des paramètres du groupe central du serveur d'applications. Les sections suivantes décrivent comment configurer les paramètres des signaux de présence du groupe central pour différentes versions de WebSphere Application Server Network Deployment :

  – **Mise à jour des paramètres des groupes centraux de WebSphere Application Server Network Deployment Version 6.x et 7.x :**

    Spécifiez l'intervalle des signaux de présence en secondes sur les versions 6.0 à 6.1.0.12 de WebSphere Application Server ou en millisecondes à partir de la version 6.1.0.13. Vous devez également spécifier le nombre de signaux de présence manqués. Cette valeur indique le nombre maximal de signaux de présence manquants avant qu'une machine virtuelle Java (JVM) ne soit considérée comme défectueuse. Le délai de détection des incidents matériels est approximativement égal au produit de l'intervalle des signaux de présence par le nombre de signaux de présence manqués.

    Ces propriétés sont spécifiées à l'aide des propriétés personnalisées sur le groupe central à l'aide de la console d'administration de WebSphere. Pour des informations de configuration détaillées, voir la rubrique Propriétés personnalisées de groupe central. Ces propriétés doivent être spécifiées pour tous les groupes centraux utilisés par l'application :

    - L'intervalle des signaux de présence est spécifié à l'aide de la propriété personnalisée IBM_CS_FD_PERIOD_SEC pour les secondes ou de la propriété personnalisée IBM_CS_FD_PERIOD_MILLIS pour les millisecondes (nécessite la version 6.1.0.13 ou une version ultérieure).
    - Le nombre de signaux de présence manqués est spécifié à l'aide de la propriété personnalisée IBM_CS_FD_CONSECUTIVE_MISSED.

La valeur par défaut de la propriété IBM_CS_FD_PERIOD_SEC est de 20 et celle de la propriété IBM_CS_FD_CONSECUTIVE_MISSED, de 10. Si la propriété IBM_CS_FD_PERIOD_MILLIS est spécifiée, elle remplace les propriétés personnalisées IBM_CS_FD_PERIOD_SEC définies. Les valeurs de ces propriétés correspondent à des entiers.

Utilisez les paramètres suivants pour spécifier un délai de détection des incidents de 1500 ms pour les serveurs WebSphere Application Server Network Deployment Version 6.x :

- Spécifiez `IBM_CS_FD_PERIOD_MILLIS = 750` (WebSphere Application Server Network Deployment V6.1.0.13 et versions ultérieures)
- Spécifiez `IBM_CS_FD_CONSECUTIVE_MISSED = 2`

– **Mise à jour des paramètres des groupes centraux de WebSphere Application Server Network Deployment Version 7.0**

WebSphere Application Server Network Deployment Version 7.0 fournit deux paramètres de groupe central qui peuvent être ajustés pour augmenter ou réduire le délai de détection des incidents :

- **Période de transmission du signal de présence.** La valeur par défaut est de 30000 millisecondes.
- **Période d'expiration du signal de présence.** La valeur par défaut est de 180000 millisecondes.

Pour plus de détails sur la manière de modifier ces paramètres, reportez-vous à la rubrique Paramètres de reconnaissance et de détection des incidents du Centre de documentation de WebSphere Application Server Network Deployment .

Utilisez les paramètres suivants pour spécifier un délai de détection des incidents de 1500 ms pour les serveurs WebSphere Application Server Network Deployment Version 7 :

- Spécifiez une période de transmission du signal de présence de 750 millisecondes.
- Spécifiez une période d'expiration du signal de présence de 1500 millisecondes.

Que faire ensuite

Lorsque vous modifiez ces paramètres pour réduire les délais de basculement, certains points d'optimisation du système sont à prendre en compte. Tout d'abord, Java n'est pas un environnement en temps réel. Des unités d'exécution peuvent être retardées si la JVM connaît des délais de récupération de place importants. Les unités d'exécution risquent également d'être retardées si la charge de la machine qui héberge la JVM est considérable (à cause de la JVM elle-même ou d'autres processus exécutés sur cette machine). Si les unités d'exécution sont retardées, les signaux de présence risquent de ne pas être envoyés à temps. Au pire, ils risquent d'être retardés du délai requis pour la reprise en ligne. Si des unités d'exécution sont retardées, des incidents sont détectés à tort. Le système doit être optimisé et dimensionné de sorte à éviter la détection de faux incidents en production. Il est recommandé pour cela de tester la charge de manière adéquate.

Remarque : La version actuelle d'eXtreme Scale prend en charge WebSphere Real Time.

Types de détection de reprise en ligne

WebSphere eXtreme Scale peut détecter les échecs de manière fiable.

Signaux de présence

1. Les sockets restent ouverts entre les machines virtuelles Java et si un socket se ferme de manière inattendue, cette fermeture est détectée comme un incident de la machine virtuelle Java homologue. Cette détection intercepte les incidents tels que l'arrêt très rapide d'une machine virtuelle Java. Une telle détection permet généralement une reprise en ligne de moins d'une seconde après ces types d'incident.
2. Les autres types d'incident incluent : un blocage du système d'exploitation, un incident physique du serveur ou une panne réseau. Ces incidents sont détectés par l'intermédiaire des signaux de présence.

Des signaux de présence sont envoyés de manière périodique entre des paires de processus : Si un nombre donné de signaux de présence sont manquants, un incident est supposé. Cette approche détecte les incidents en N*M secondes, N représentant le nombre de signaux de présence manquants et M, l'intervalle auquel les signaux de présence doivent être définis. Il n'est pas possible de spécifier directement M et N ; à la place, un mécanisme de règle est utilisé pour autoriser une plage de combinaisons M et N testées à utiliser.

Echecs

Il existe plusieurs types d'échecs des processus. Un processus peut échouer car la limite des ressources (par exemple la taille maximale des segments de mémoire) a été atteinte ou parce qu'une logique de contrôle de processus a mis fin à un processus. Ceci risque alors d'entraîner un échec du système d'exploitation et la perte des processus en cours d'exécution. Moins fréquemment, une défaillance du matériel, par exemple de la carte d'interface réseau, peut se produire : le système d'exploitation est alors déconnecté du réseau. Dans ce contexte, les échecs peuvent être classés en deux types : échec de processus et perte de connectivité.

Echec de processus

WebSphere eXtreme Scale réagit très rapidement à un échec de processus. Lorsqu'un processus échoue, le système d'exploitation est responsable du nettoyage des ressources utilisées par celui-ci. Ce nettoyage comprend l'allocation de port et la connectivité. Lorsqu'un processus échoue, un signal est immédiatement envoyé via les connexions utilisées par ce processus pour fermer celles-ci.

Perte de connectivité

Une perte de connectivité se produit lorsque le système d'exploitation est déconnecté. Il ne parvient alors plus à envoyer des signaux à d'autres processus. Plusieurs raisons peuvent expliquer la perte de connectivité. Elles peuvent être classées en deux catégories : échec de l'hôte ou îlotage.

Echec de l'hôte

Si une machine hôte n'est plus alimentée, elle devient instantanément non disponible.

Ilotage

Ce scénario constitue l'échec le plus complexe à gérer par le logiciel car le processus est censé être indisponible, alors qu'il ne l'est pas.

Echec de conteneur

L'échec d'un conteneur est généralement identifié par des conteneurs homologues via le mécanisme du groupe central. Lorsqu'un conteneur ou un jeu de conteneurs échoue, le service de catalogue migre les fragments hébergés par ce ou ces conteneurs. Le service de catalogue commence par rechercher un fragment réplique synchrone avant de procéder à la migration vers un fragment réplique asynchrone. Une fois les fragments primaires migrés vers les nouveaux conteneurs, le service de catalogue recherche de nouveaux conteneurs hôtes pour les fragments répliques manquants.

Remarque : Ilotage de conteneurs - Le service de catalogue fait migrer les fragments de certains conteneurs lorsque ceux-ci s'avèrent indisponibles. S'ils redeviennent disponibles, le service de catalogue considèrent que des fragments peuvent à nouveau y être positionnés, comme dans le flux de démarrage normal.

Temps d'attente de détection des basculements

Les échecs peuvent être classés en deux catégories : les échecs liés aux logiciels et les échecs liés au matériel. Les échecs liés aux logiciels se produisent généralement lorsqu'un processus échoue. Ils sont détectés par le système d'exploitation, qui parvient très rapidement à récupérer les ressources utilisées, par exemple les sockets réseau. Cette détection se fait en général en moins d'une seconde. La détection des échecs liés au matériel se fait à l'aide de la fonction d'optimisation du signal de présence par défaut. Elle peut prendre jusqu'à 200 secondes. Ces échecs peuvent prendre la forme suivante : panne d'une machine physique, déconnexion d'un câble réseau ou échec du système d'exploitation. eXtreme Scale se fie donc au signal de présence pour détecter les échecs liés au matériel qui peuvent être configurés.

Echec de plusieurs conteneurs

Un fragment réplique n'est jamais placé dans le même processus que le fragment primaire, car en cas de perte du processus, les deux fragments seraient perdus. La règle de déploiement définit un attribut utilisé par le service de catalogue pour déterminer si un fragment réplique doit être placé sur la même machine que le fragment primaire. Dans un environnement de développement ne comprenant qu'une machine, vous pouvez prévoir deux conteneurs afin de procéder à des fragments répliques. Dans un environnement de production, l'utilisation d'une seule machine est cependant insuffisante car une éventuelle perte de cette machine hôte entraînerait la perte des deux conteneurs. Pour passer d'un environnement de développement comptant une seule machine à un environnement de production comprenant plusieurs machines, désactivez le mode de développement dans le fichier de configuration de la règle de déploiement.

Echec du service de catalogue

La grille du service de catalogue étant aussi une grille eXtreme Scale, elle utilise le mécanisme de regroupement central de la même manière que le processus d'échec des conteneurs, à cette différence près : le domaine de services de catalogue utilise une sélection d'homologue pour définir le fragment primaire plutôt que l'algorithme de service de catalogue utilisé pour les conteneurs.

Il est à noter que le service de positionnement et le service de regroupement central sont des services Un sur N. Un service Un sur N ne s'exécute que sur un seul membre du groupe haute disponibilité. Le service de localisation et

l'administration s'exécutent, quant à eux, sur tous les membres de ce groupe. Le service de positionnement et le service de regroupement central sont des singletons car ils sont responsables de l'agencement du système. Le service de localisation et d'administration sont des services en lecture seule qui existent partout à des fins d'évolutivité.

Le service de catalogue utilise la réplication pour garantir sa tolérance aux erreurs. Si un processus de service de catalogue échoue, il doit redémarrer pour rétablir le niveau de disponibilité du système. Si tous les processus hébergeant le service de catalogue échouent, des données critiques sont perdues. Vous devez alors impérativement redémarrer tous les conteneurs. Comme le service de catalogue peut s'exécuter sur plusieurs processus, cet échec est improbable. Toutefois, si vous exécutez tous les processus sur un même sous-système de stockage, sur un même chassis lame ou à partir d'un même commutateur réseau, un échec est très probable. Essayez de supprimer les modes d'échecs les plus connus des sous-systèmes de stockage qui hébergent le service de catalogue pour limiter les risques d'échec.

Tableau 29. Récapitulatif de la reconnaissance d'échec et de la reprise en ligne

Type de perte	**Mécanisme de reconnaissance (détection)**	**Méthode de récupération**
Perte de processus	E-S	Redémarrage
Perte de serveur	Signal de présence	Redémarrage
Indisponibilité du réseau	Signal de présence	Rétablissement du réseau et de la connexion
Blocage côté serveur	Signal de présence	Arrêt et redémarrage du serveur
Serveur occupé	Signal de présence	Attendre que le serveur soit disponible

Utilisation de WebSphere Real Time

Utiliser WebSphere eXtreme Scale avec WebSphere Real Time augmente la cohérence et la prévisibilité des performances de débit qu'offre la stratégie standard de récupération de place employée dans le JRE (IBM Java™ SE Runtime Environment). Le ratio coûts/avantages est variable. WebSphere eXtreme Scale crée un grand nombre d'objets temporaires associés à chaque transaction. Ces objets temporaires s'occupent des demandes, des réponses, des séquences de journaux et des sessions. Sans WebSphere Real Time, les temps de réponse des transactions peuvent grimper à des centaines de millisecondes. Mais WebSphere Real Time utilisé avec WebSphere eXtreme Scale peut augmenter l'efficacité de la récupération de place et faire tomber à 10% les temps de réponse de la configuration autonome.

WebSphere Real Time en environnement autonome

Il est possible d'utiliser WebSphere Real Time avec WebSphere eXtreme Scale. En activant WebSphere Real Time, l'on obtient une récupération de place plus prévisible grâce à des temps de réponses stables et cohérents et des débits de transactions dans un environnement eXtreme Scale autonome.

Avantages de WebSphere Real Time

WebSphere eXtreme Scale crée un grand nombre d'objets temporaires associés à chaque transaction. Ces objets temporaires s'occupent des demandes, des réponses, des séquences de journaux et des sessions. Sans WebSphere Real Time, les temps de réponse des transactions peuvent grimper à des centaines de millisecondes. Mais WebSphere Real Time utilisé avec WebSphere eXtreme Scale peut augmenter l'efficacité de la récupération de place et faire tomber à 10% les temps de réponse de la configuration autonome.

Activer WebSphere Real Time

Installez WebSphere Real Time et WebSphere eXtreme Scale autonome sur les ordinateurs sur lesquels vous prévoyez d'exécuter eXtreme Scale. Définissez la variable d'environnement JAVA_HOME pour qu'elle pointe sur un JRE standard Java SE standard.

Définissez la variable d'environnement JAVA_HOME pour qu'elle pointe sur le uWebSphere Real installé. Puis activez WebSphere Real Time comme indiqué ci-après.

1. Dans le fichier `objectgridRoot/bin/setupCmdLine.sh | .bat`, supprimez le commentaire de la ligne suivante :

   ```
   WXS_REAL_TIME_JAVA="-Xrealtime -Xgcpolicy:metronome
   -Xgc:targetUtilization=80"
   ```

2. Enregistrez le fichier.

WebSphere Real Time est à présent activé. Pour le désactiver, il vous suffit de repasser la même ligne en commentaire.

Meilleures pratiques

WebSphere Real Time confère aux transactions eXtreme Scale des temps de réponse plus prévisibles. Les résultats montrent que les temps de réponse d'une transaction eXtreme Scale s'améliorent de manière significative avec WebSphere Real Time si on les compare à ceux obtenus par le récupérateur de place par défaut de Java. L'activation de WebSphere Real Time avec eXtreme Scale est un must si la stabilité et les temps de réponse de votre application sont essentiels.

Les pratiques recommandées développées ci-après expliquent comment rendre WebSphere eXtreme Scale encore plus efficace grâce à une optimisation et une programmation fonction de la charge attendue.

- Définissez le bon niveau d'utilisation du processeur et de récupération de place.

  WebSphere Real Time donne les moyens de contrôler l'utilisation du processeur de manière à contrôler et à réduire l'impact de la récupération de place sur votre application. Le paramètre `-Xgc:targetUtilization=NN` permet de spécifier `NN` comme pourcentage du processeur qui est utilisé par votre application toutes les 20 secondes. Par défaut, cette valeur est de 80 % pour WebSphere eXtreme Scale, mais vous pouvez modifier le script dans le fichier `objectgridRoot/bin/setupCmdLine.sh` pour définir un autre chiffre, `70`, par exemple, qui libère davantage de capacité processeur pour le récupérateur de place. Déployez suffisamment de serveurs pour maintenir la charge processeur en dessous de 80 % pour vos applications.
- Augmentez la taille de la mémoire dynamique.

WebSphere Real Time utilise davantage de mémoire que le Java standard, aussi, prévoyez plus de mémoire dynamique pour votre WebSphere eXtreme Scale et définissez la taille du segment mémoire lors du démarrage des serveurs de catalogue avec le paramètre `–jvmArgs –XmxNNNM` dans la commande ogStartServer. Vous pouvez, par exemple, utiliser le paramètre `–jvmArgs –Xmx500M` pour démarrer des serveurs de catalogue et utilisez une taille mémoire appropriée pour démarrer les conteneurs. Vous pouvez fixer la taille de la mémoire à 60-70 % de la taille prévue par machine virtuelle Java pour vos données . Si vous ne définissez pas cette valeur, une erreur OutOfMemoryError risque de se produire. Vous pouvez également, si vous le souhaitez, utiliser le paramètre `–jvmArgs –Xgc:noSynchronousGCOnOOM` pour empêcher le comportement non déterministe lorsque la machine virtuelle Java est à court de mémoire.

- Ajustez les unités d'exécution pour la récupération de place.

  WebSphere eXtreme Scale crée un grand nombre d'objets temporaires associés à chaque transaction et aux unités d'exécution RPC (Remote Procedure Call). La récupération de place présente des avantages pour les performances si votre ordinateur dispose de suffisamment de cycles processeur. Le nombre d'unités d'exécution est de 1 par défaut. L'argument `–Xgcthreads n` permet de modifier ce nombre. La valeur suggérée pour cet argument est le nombre de coeurs qui sont disponibles en prenant en considération le nombre de machines virtuelles Java par ordinateur.

- Ajustez les performances pour les applications à exécution courte avec WebSphere eXtreme Scale.

  WebSphere Real Time est optimisé pour les applications à exécution longue. D'ordinaire, vous avez besoin d'exécuter des transactions WebSphere eXtreme Scale pendant deux heures en continu pour obtenir des données de performances fiables. Le paramètre `–Xquickstart` donne de meilleures performances à vos applications à exécution courte. Ce paramètre indique au compilateur JIT (just-in-time) d'utiliser un bas niveau d'optimisation.

- Réduisez la file d'attente des clients WebSphere eXtreme Scale et les relais clients WebSphere eXtreme Scale.

  Le principal avantage d'utiliser WebSphere eXtreme Scale avec WebSphere Real Time est de bénéficier de temps de réponse des transactions extrêmement fiables, qui représentent usuellement une amélioration de l'ordre de plusieurs fois l'écart constaté dans les temps de réponse des transactions. Toutes les demandes clients mises en file d'attente et tous les relais de ces demandes effectuées via d'autres logiciels ont un impact sur les temps de réponse qui échappent au contrôle de WebSphere Real Time et de WebSphere eXtreme Scale. Vous devez modifier les paramètres de vos unités d'exécution et de vos sockets pour conserver une charge à la fois ferme et fluide sans retards significatifs et vous devez diminuer la profondeur des files d'attente.

- Ecrivez des applications WebSphere eXtreme Scale qui utilisent les unités d'exécution WebSphere Real Time.

  Sans modifier votre application, vous pouvez obtenir des temps de réponse WebSphere eXtreme Scale des transactions extrêmement fiable représentant une amélioration de l'ordre de plusieurs fois l'écart standard dans les temps de réponse des transactions. Vous pouvez exploiter davantage les unités d'exécution de vos applications transactionnelles en passant du threading Java standard au RealtimeThread, qui fournit un meilleur contrôle de la priorité des unités d'exécution et de la planification.

  Actuellement, votre application comporte le code suivant :

  ```
  public class WXSCacheAppImpl extends Thread implements WXSCacheAppIF
  ```

Vous pouvez remplacer ce code par le code suivant :

```
public class WXSCacheAppImpl extends RealtimeThread implements
WXSCacheAppIF
```

WebSphere Real Time sur WebSphere Application Server

Vous pouvez utiliser WebSphere® Real Time avec eXtreme Scale dans un environnement WebSphere Application Server Network Deployment de version 7.0. L'activation de WebSphere Real Time permet d'obtenir une récupération de place plus prévisible avec des temps de réponses et des débits de transactions stables et cohérents.

Avantages

Utiliser WebSphere eXtreme Scale avec WebSphere Real Time augmente la cohérence et la prévisibilité des performances de débit qu'offre la stratégie standard de récupération de place employée dans le JRE (IBM Java™ SE Runtime Environment). Le ratio coûts/avantages est variable en fonction de plusieurs critères. Voici quelques-uns des principaux critères :

- capacités en serveurs : mémoire disponible, vitesse et taille des processeurs, vitesse et utilisation du réseau
- charges des serveurs : charge processeur soutenue, charge processeur de pointe
- configuration Java : taille des segments, utilisation cible, unités d'exécution de récupération de place
- configuration du mode copie de WebSphere eXtreme Scale : tableau d'octets ou stockage POJO
- points propres aux applications : utilisation des unités d'exécution, conditions requises et tolérance des réponses, taille des objets, etc.

En plus de la stratégie métronome de récupération de place utilisable dans WebSphere Real Time, il existe des stratégies optionnelles proposées par le JRE IBM standard. Ces stratégies, optthruput (stratégie par défaut), gencon, optavgpause et subpool sont spécifiquement conçues pour résoudre les différents besoins et environnements des applications. Pour plus d'informations sur ces stratégies, voir «Optimisation des machines virtuelles Java pour WebSphere eXtreme Scale», à la page 429. En fonction des besoins de l'application et de l'environnement, ainsi que des ressources et des restrictions, le prototypage d'une ou plusieurs de ces stratégies peut vous garantir la satisfaction de ces besoins et vous aider à déterminer à coup sûr une stratégie optimale.

Possibilités avec WebSphere Application Server Network Deployment

1. Voici quelques-unes des versions prises en charge :
   - WebSphere Application Server Network Deployment version 7.0.0.5 et au-dessus
   - WebSphere Real Time V2 SR2 for Linux et au-dessus. Pour plus d'informations, voir IBM WebSphere Real Time V2 for Linux
   - WebSphere eXtreme Scale version 7.0.0.0 et au-dessus
   - Linux 32 et 64 bits
2. Les serveurs WebSphere eXtreme Scale ne peuvent cohabiter avec WebSphere Application Server DMgr.
3. Real Time ne prend pas en charge DMgr.
4. Real Time ne prend pas en charge les agents de noeuds WebSphere.

Activer WebSphere Real Time

Installez WebSphere Real Time et WebSphere eXtreme Scale sur les ordinateurs sur lesquels vous prévoyez d'exécuter eXtreme Scale. Mettez au niveau SR2 le Java de WebSphere Real Time.

Vous pouvez spécifier comme suit les paramètres des machines virtuelles Java pour chaque serveur via la console WebSphere Application Server version 7.0.

Sélectionnez **Serveurs → Types de serveur → Serveurs d'applications WebSphere → <serveur installé requis>**.

Dans la page qui s'affiche, choisissez Définition des processus.

Dans la page qui s'affiche alors, cliquez sur Machine virtuelle Java en haut de la colonne de droite (c'est là que vous pouvez définir pour chaque serveur la taille des segments, la récupération de place et d'autres indicateurs).

Définissez les indicateurs suivants dans la zone Arguments JVM génériques :

```
-Xrealtime -Xgcpolicy:metronome  -Xnocompressedrefs -Xgc:targetUtilization=80
```

Appliquez les modifications et enregistrez-les.

Pour utiliser Real Time dans WebSphere Application Server 7.0 avec des serveurs eXtreme Scale incluant les indicateurs JVM ci-dessus, vous devez créer une variable d'environnement JAVA_HOME.

Définissez JAVA_HOME comme suit :

1. Développez Environnement.
2. Sélectionnez Variables WebSphere.
3. La case Toutes les portées en dessous de Afficher la portée doit être cochée.
4. Sélectionnez le serveur requis dans la liste déroulante (ne sélectionnez pas de serveurs DMgr ou d'agents de noeuds).
5. Si la variable d'environnement JAVA_HOME n'apparaît pas dans la liste, sélectionnez Nouveau et spécifiez JAVA_HOME comme nom de la variable. Dans la zone Valeur, entrez le nom complet du chemin d'accès à Real Time.
6. Appliquez les modifications et enregistrez-les.

Meilleures pratiques

Vous trouverez un ensemble de pratiques recommandées dans la section Meilleures pratiques du chapitre «Utilisation de WebSphere Real Time», à la page 438. Il y a dans cette liste de pratiques recommandées pour un environnement WebSphere eXtreme Scale autonome des points qui diffèrent pour un déploiement dans un environnement WebSphere Application Server Network Deployment.

Vous devez placer des paramètres supplémentaires de ligne de commande JVM au même endroit que les paramètres de stratégie de récupération de place évoqués à la précédente section.

Une cible initiale acceptable pour les charges processeur soutenues est de 50 % avec des charges de pointe de courte durée grimpant jusqu'à 75 %. Au-delà, vous devez ajouter des capacités supplémentaires avant de constater une dégradation mesure de la prévisibilité et de la cohérence. Vous pouvez augmenter légèrement

les performances si vous pouvez tolérer des temps de réponse plus longs. Au-delà d'un seuil de 80 % conduit souvent à une dégradation significative de la cohérence et de la prévisibilité.

Optimisation du fournisseur de cache dynamique

Le fournisseur de cache dynamique de WebSphere eXtreme Scale prend en charge les paramètres de configuration ci-après pour l'optimisation des performances.

Pourquoi et quand exécuter cette tâche

- **com.ibm.websphere.xs.dynacache.ignore_value_in_change_event** : lorsque vous enregistrez un programme d'écoute d'événements de modification auprès du fournisseur de cache dynamique et que vous générez une instance ChangeEvent, une charge supplémentaire est associée à la désérialisation de l'entrée de cache pour que la valeur puisse être placée dans l'événement de modification. Si vous affectez la valeur `true` à ce paramètre facultatif dans l'instance de cache, la désérialisation de l'entrée de cache est ignorée lors de la génération des événements de modification. La valeur renvoyée est null dans le cas d'une opération de suppression ou correspond à un tableau d'octets contenant la forme sérialisée de l'objet. Les instances InvalidationEvent font l'objet d'une pénalité de performances similaire, que vous pouvez éviter en affectant à com.ibm.ws.cache.CacheConfig.ignoreValueInInvalidationEvent la valeur `true`.
- **com.ibm.websphere.xs.dynacache.enable_compression**: par défaut, le fournisseur de cache dynamique d'eXtreme Scale compresse en mémoire les entrées du cache pour augmenter la densité de ce dernier, ce qui peut faire gagner une quantité importante de mémoire aux applications comme la mise en cache de servlets. Si vous savez que la plupart de vos données de cache ne seront pas compressibles, vous pouvez définir cette valeur comme false.

Optimiser l'agent de dimensionnement du cache pour des estimations précises de l'utilisation de la mémoire

A partir de la version 7.1, WebSphere eXtreme Scale prend en charge le dimensionnement de l'utilisation de la mémoire par les mappes de sauvegarde dans les grilles réparties (ce dimensionnement n'est pas pris en charge pour les instances de grilles locales). Dans la plupart des cas, la valeur signalée par WebSphere eXtreme Scale pour une mappe donnée est très proche de celle signalée par l'analyse des clichés de segments mémoire. La complexité de l'objet mappe peut néanmoins entraver la précision du dimensionnement. En fait, le message CWOBJ4543 s'affiche dans le journal pour n'importe quelle objet d'entrée qui ne peut être dimensionné avec précision en raison de sa complexité. L'observation des pratiques recommandées ici pour éviter aux mappes une complexité superflue renforcera la précision des dimensionnements.

Procédure

- Activez l'agent de dimensionnement.

  Avec une machine virtuelle Java 5 ou supérieure, vous bénéficiez d'un agent de dimensionnement qui permet à WebSphere eXtreme Scale d'obtenir de la machine virtuelle Java des informations supplémentaires qui amélioreront la précision de ses estimations. Il est possible de charger l'agent en ajoutant l'argument suivant à la ligne de commande de la machine virtuelle Java :

  `-javaagent:`*`rép_biblio_WXS`*`/wxssizeagent.jar`

  Dans le cas d'une topologie imbriquée, ajoutez l'argument à la ligne de commande du processus WebSphere Application Server.

Dans le cas d'une topologie répartie, ajoutez l'argument à la ligne de commande des processus eXtreme Scale (conteneurs) et du processus WebSphere Application Server.

Lorsque le chargement s'est effectué correctement, le message suivant est écrit dans le fichier `SystemOut.log` :

```
CWOBJ4541I: La fonction de définition optimisée de la taille de la mémoire
BackingMap est activée.
```

- Dans la mesure du possible, utilisez de préférence des types de données Java plutôt que des types de données personnalisés.

  WebSphere eXtreme Scale sait estimer avec précision le coût en mémoire des types suivants :

  – java.lang.String et les tableaux où String est la classe de composant (String[])
  – tous les types d'encapsuleurs primitifs (Byte, Short, Character, Boolean, Long, Double, Float, Integer) et les tableaux où ces encapsuleurs primitifs sont le type de composant (par exemple, Integer[], Character[])
  – java.math.BigDecimal et java.math.BigInteger ainsi que les tableaux où ces deux classes sont le type de composant (BigInteger[] et BigDecimal[])
  – les types temporels (java.util.Date, java.sql.Date, java.util.Time, java.sql.Timestamp)
  – java.util.Calendar et java.util.GregorianCalendar

- Dans la mesure du possible, évitez le confinement d'objet.

  Lorsqu'un objet est inséré dans une mappe, WebSphere eXtreme Scale part du principe qu'il est le seul à détenir une référence à l'objet et à tous les autres objets directement référencés par cet objet. Si vous insérez mille objets personnalisés dans une mappe et que chacun de ces objets comporte une référence à la même instance String, WebSphere eXtreme Scale dimensionnera mille fois cette instance, surestimant de ce fait la taille réelle de la mappe dans le segment de mémoire. Cela dit, WebSphere eXtreme Scale procédera à une compensation correcte pour ces scénarios usuels de confinement :

  – références aux énums Java 5
  – références aux classes conformes au pattern Typesafe Enum. Les classes conformes à ce pattern n'auront en effet que des constructeurs privés définis. Elles auront au moins un champ final statique privé de son propre type et, si elles implémentent Serializable, elles implémenteront readResolve()
  – confinement d'un encapsuleur primitif Java 5. Utilisation par exemple d'Integer.valueOf(1) au lieu du nouvel Integer(1)

  En conséquence de quoi, si vous devez procéder à un confinement, utilisez l'une des techniques qui précèdent.

- Utilisez les types personnalisés à bon escient.

  Lorsque vous devez utiliser des types personnalisés, privilégiez les types primitifs de données pour les champs, plutôt que les types d'objets.

  Préférez également à vos propres implémentations personnalisées les types d'objets répertoriés dans l'entrée 2.

  Lorsque vous utilisez des types personnalisés, veillez à ce que l'arborescence des objets n'ait pas plus d'un niveau. Lorsqu'on insère un objet personnalisé dans une mappe, WebSphere eXtreme Scale ne calculera que le coût de l'objet inséré, ce qui inclut tous les champs primitifs et tous les objets directement référencés par l'objet inséré. WebSphere eXtreme Scale n'ira pas plus loin pour suivre les références dans l'arborescence. Si vous insérez un objet dans la mappe et que WebSphere eXtreme Scale détecte des références qui n'ont pas été suivies lors du dimensionnement, vous vous retrouverez avec le message CWOBJ4543 qui

inclura le nom de la classe qui n'a pu être entièrement dimensionnée. Lorsque cela se produit, traitez les statistiques de taille de la mappe comme des données tendancielles au lieu de les considérer comme un total exact.

- Si possible, utilisez CopyMode.COPY_TO_BYTES.

  CopyMode.COPY_TO_BYTES supprime toute incertitude du dimensionnement des objets valeur insérés dans la mappe même lorsqu'une arborescence d'objets comporte trop de niveaux pour être dimensionnée normalement (ce qui se traduit par le fameux message CWOBJ4543).

Définir la taille du cache en fonction de son utilisation

A partir de sa version 7.1, WebSphere eXtreme Scale sait estimer avec précision l'utilisation en octets de la mémoire dynamique Java par une mappe de sauvegarde donnée. Cette capacité vous permet de dimensionner correctement les segments de mémoire de vos machines virtuelles Java et de définir de manière appropriée les règles d'expulsion. Le comportement de cette évaluation varie selon le degré de complexité des objets placés dans la mappe de sauvegarde et selon la configuration de la mappe. A l'heure actuelle, cette fonctionnalité n'est prise en charge que pour les grilles réparties. Il n'est pas possible d'évaluer le nombre d'octets utilisés par les instances de grilles locales.

eXtreme Scale stocke toutes ses données dans les segments mémoire des processus de machines virtuelles Java constituant la grille. Pour une mappe donnée, le segment mémoire qu'utilise cette mappe peut se décomposer dans les éléments suivants :

- la taille de tous les objets Key actuellement dans la mappe
- la taille de tous les objets Value actuellement dans la mappe
- la taille de tous les objets EvictorData utilisés par les plug-in Evictor dans cette mappe
- la taille de la structure de données sous-jacente

Le nombre d'octets utilisés retourné par les statistiques de dimensionnement sera donc la somme de ces quatre composants. Ces valeurs sont calculés entrée par entrée dans les opérations d'insertion, d'actualisation et de suppression sur la mappe, ce qui signifie qu'eXtreme Scale sait en permanence le nombre d'octets utilisés par une mappe de sauvegarde donnée.

Chaque partition d'une grille partitionnée contient un morceau de la mappe de sauvegarde. Les statistiques de dimension étant calculées au plus bas niveau du code d'eXtreme Scale, chaque partition d'une mappe de sauvegarde effectue le suivi de sa propre taille. Vous pouvez utiliser les API Statistics d'eXtreme Scale pour suivre la taille cumulée de l'ensemble de la mappe ainsi, d'ailleurs, que les tailles individuelles des partitions qui composent cette mappe.

De manière générale, les données de taille doivent être traitées comme des données indicatives de tendances, par opposition à la mesure exacte de l'espace mémoire utilisé par la mappe. Ainsi, si la taille signalée d'une mappe double de 5 à 10 Mo, considérez que l'utilisation de la mémoire par la mappe a doublé. Ce chiffre de 10 Mo peut très bien être inexact pour un certain nombre de raisons dans le détail desquelles nous allons rentrer. Si vous tenez compte de ces raisons et que vous appliquez les bonnes pratiques que nous préconisons, le dimensionnement sera presque aussi exact que le post-traitement d'un cliché de segment mémoire Java.

Le principal problème avec l'exactitude vient de ce que le modèle mémoire de Java n'est pas suffisamment restrictif et qu'il tolère une certaine marge de précision dans

les mesures de la mémoire. Le problème fondamental est qu'un objet peut continuer à être présent dans le segment mémoire pour cause de références multiples. C'est le cas, par exemple, d'une instance d'objet de 5 Ko qui est insérée dans trois mappes distinctes. Dans ce cas, dans aucune de ces trois mappes, l'objet ne pourra être éliminé par la récupération de place. Dans ce cas de figure, les mesures suivantes sont tout aussi valables :

- la taille de chaque mappe s'agrandit de 5 Ko
- la taille de la première mappe dans laquelle est positionné l'objet s'agrandit de 5 Ko
- les deux autres mappes ne sont pas agrandies. La taille de chacune des mappes est augmentée d'une fraction de la taille de l'objet

C'est en raison de cette ambiguïté que ces mesures doivent être prises comme indicatives sauf à avoir supprimé l'ambiguïté grâce à des choix de conception, des pratiques recommandées et une compréhension des choix d'implémentation effectuées avec cette fonctionnalité.

eXtreme Scale présuppose que chaque mappe est la seule à détenir la référence durable aux objets Key et Value qu'elle contient. De ce fait, si le même objet de 5 Ko est placé dans trois mappes, la taille de chacune de ces mappes s'agrandira de 5 Ko. Cette augmentation n'est d'ordinaire pas un problème, car la fonctionnalité ne concerne que les grilles réparties. Si vous insérez le même objet dans trois mappes différentes sur un client distant, chaque mappe aura sa propre copie de l'objet. Par ailleurs, les paramètres par défaut du mode transactionnel COPY MODE garantissent que chaque mappe ait sa propre copie d'un objet donné.

Le confinement des objets sera le plus gros défi pour la plupart des applications des clients. Il y a confinement des objets lorsque le code de l'application fait exprès en sorte que toutes les références à la valeur d'un objet donné pointent effectivement sur la même instance de l'objet dans le segment de mémoire. Exemple : la classe suivante.

```
public class ShippingOrder implements Serializeable,Cloneable{

    public static final STATE_NEW = "new";
    public static final STATE_PROCESSING = "processing";
    public static final STATE_SHIPPED = "shipped";

    private String state;
    private int orderNumber;
private int customerNumber;

public Object clone(){
       ShippingOrder toReturn = new ShippingOrder();
       toReturn.state = this.state;
       toReturn.orderNumber = this.orderNumber;
       toReturn.customerNumber = this.customerNumber;
       return toReturn;
    }

    private void readResolve(){
        if (this.state.equalsIgnoreCase("new")
            this.state = STATE_NEW;
        else if (this.state.equalsIgnoreCase("processing")
            this.state = STATE_PROCESSING;
        else if (this.state.equalsIgnoreCase("shipped")
            this.state = STATE_SHIPPED:
    }
}
```

Quelle que soit la configuration d'eXtreme Scale, le coût réel de cette classe sera surestimé par les statistiques de dimensionnement. S'il existe un million d'objets ShippingOrder, par nature, le dimensionnement reflétera le coût d'un million de chaînes détenant les informations state, alors que, en réalité, il n'y a que trois chaînes et que ce sont des membres d'une classe statique. Le coût en mémoire ne devrait jamais être ajoutée à une mappe eXtreme Scale, mais ce n'est pas le bon moyen de détecter cette situation lors de l'exécution. Or, il existe des dizaines de manière de réaliser un confinement de ce genre, et c'est bien pour cela que c'est aussi ardu à détecter. eXtreme Scale n'a pas les moyens de se protéger contre la totalité de ces moyens. Néanmoins, il arrive à se protéger contre les types les plus communément utilisés.

eXtreme Scale s'ajustera automatiquement pour les énumérations Java 5 et le pattern Typesafe Enum (voir http://java.sun.com/j2se/1.5.0/docs/guide/language/enums.html).

Pour optimiser l'utilisation de la mémoire grâce au confinement d'objets, ne confinez que les objets personnalisés qui relèvent de ces deux catégories. L'exactitude des statistiques d'utilisation de la mémoire ne pourra que s'en trouver renforcée. En outre, Java 5 a vu l'introduction d'un confinement automatique pour les types primitifs d'encapsuleurs (Integer, par exemple) via le recours aux méthodes statiques valueOf(). eXtreme Scale prendra automatiquement en compte ce confinement.

Pour accéder aux statistiques d'utilisation de la mémoire, utilisez l'une des méthodes suivantes.

API Statistics

La méthode MapStatsModule.getUsedBytes() fournit les statistiques d'une mappe individuelle (nombre d'entrées et taux de réussite).

Pour des détails, voir «Modules des statistiques», à la page 386.

Beans gérés (MBeans)

Utilisez la statistique MapUsedBytes des beans gérés. Vous pouvez utiliser plusieurs types de beans gérés JMX (Java Management Extensions) différents pour administrer et surveiller les déploiements. Chaque bean géré fait référence à une entité spécifique, telle qu'une mappe, eXtreme Scale, un serveur, un groupe de réplication ou un membre de groupe de réplication.

Pour des détails, voir «Administration par programmation à l'aide de beans gérés (MBeans)», à la page 355.

Modules PMI

Vous pouvez surveiller les performances de vos applications avec les modules PMI. Particulièrement, utilisez le module PMI de mappe pour les conteneurs imbriqués dans WebSphere Application Server.

Pour des détails, voir «Modules PMI», à la page 394.

Console WebSphere eXtreme Scale

La console, qui a fait son apparition avec la version 7.1, permet de visualiser les statistiques d'utilisation de la mémoire. Voir «Surveillance à l'aide de la console Web», à la page 403.

Toutes ces méthodes accèdent à la base aux mêmes mesures de l'utilisation de la mémoire par une instance BaseMap donnée. L'environnement d'exécution de WebSphere eXtreme Scale tente (meilleur effort) de calculer le nombre d'octets de la mémoire dynamique qui sont utilisées par les objets Key et Value stockés dans la mappe, ainsi que le coût total de la mappe elle-même. Vous pouvez voir combien de mémoire dynamique chaque mappe utilise dans l'ensemble de la grille répartie.

Dans la plupart des cas, la valeur signalée par WebSphere eXtreme Scale pour une mappe donnée sera très proche de celle signalée par l'analyse des clichés de segments mémoire. WebSphere eXtreme Scale évaluera avec précision sa taille d'ensemble, sans pouvoir rendre compte de manière précise de chaque objet ayant pu être placé dans les mappes. L'observation des pratiques recommandées dans «Optimiser l'agent de dimensionnement du cache pour des estimations précises de l'utilisation de la mémoire», à la page 443 renforcera la précision des dimensionnements en octets fournis par WebSphere eXtreme Scale.

Chapitre 11. Résolution des incidents

En plus des journaux et de la trace ainsi que des des messages et des notes sur l'édition, mentionnés dans la présente section, vous pouvez utiliser les outils de surveillance pour identifier et résoudre les incidents liés à l'emplacement des données dans l'environnement, la disponibilité des serveurs dans la grille, etc. Si vous utilisez un environnement WebSphere Application Server, vous pouvez utiliser PMI (Performance Monitoring Infrastructure). Si vous utilisez un environnement autonome, vous pouvez utiliser l'outil de surveillance d'un fournisseur, tel que CA Wily Introscope ou Hyperic HQ. Vous pouvez également utiliser et personnaliser l'exemple d'utilitaire xsAdmin pour afficher des informations textuelles sur votre environnement.

Journaux et trace

Vous pouvez utiliser des journaux et une trace pour surveiller votre environnement et en identifier et résoudre les problèmes. Les journaux se trouvent dans des emplacements différents suivant votre configuration. Il se peut que vous deviez fournir une trace pour un serveur lorsque vous travaillez avec le service d'assistance d'IBM.

Journaux avec WebSphere Application Server

Pour plus d'informations, voir le centre de documentation de WebSphere Application Server.

Journaux avec WebSphere eXtreme Scale dans un environnement autonome

Avec les serveurs de catalogues et de conteneurs autonomes, vous définissez l'emplacement des journaux et des spécifications de trace. Les journaux des serveurs de catalogues se trouvent dans l'emplacement où vous avez exécuté la commande de démarrage des serveurs.

Définition de l'emplacement des journaux des serveurs conteneurs

Par défaut, les journaux d'un conteneur se trouvent dans le répertoire où la commande serveur a été exécutée. Si vous démarrez les serveurs dans le répertoire *<rép_base_eXtremeScale>*/bin, les journaux et les fichiers de trace se trouvent dans les répertoires logs/*<nom_serveur>* du répertoire bin. Pour spécifier un autre emplacement pour les journaux des serveurs conteneurs, créez un fichier de propriétés, tel que le fichier server.properties, avec le contenu suivant :

```
workingDirectory=<directory>
traceSpec=
systemStreamToFileEnabled=true
```

La propriété workingDirectory correspond au répertoire racine des journaux et du fichier de trace facultatif. WebSphere eXtreme Scale crée un répertoire du nom du serveur conteneur avec un fichier SystemOut.log, un fichier SystemErr.log et un fichier de trace si la trace a été activée avec l'option traceSpec. Pour utiliser un fichier de propriétés au démarrage des conteneurs, utilisez l'option **-serverProps** et spécifiez l'emplacement du fichier de propriétés du serveur.

Pour plus d'informations, reportez-vous aux rubriques «Démarrage de serveurs WebSphere eXtreme Scale autonomes», à la page 330 et «Script startOgServer», à la page 336.

Les messages d'information courants à rechercher dans le fichier `SystemOut.log` sont les messages de confirmation du démarrage. Pour plus d'informations sur un message spécifique, voir «Messages», à la page 454.

Trace avec WebSphere Application Server

Pour plus d'informations, voir le centre de documentation de WebSphere Application Server.

Trace sur le service de catalogue

Vous pouvez définir la trace sur un service de catalogue à l'aide des paramètres **-traceSpec** et **-traceFile** au démarrage du service de catalogue. Par exemple :

```
startOgServer.sh catalogServer -traceSpec
ObjectGridPlacement=all=enabled -traceFile
/home/user1/logs/trace.log
```

Si vous démarrez le service de catalogue dans le répertoire *<rép_base_eXtremeScale>*`/bin`, les journaux et les fichiers de trace se trouveront dans le répertoire `logs/`*<nom_service_catalogue>* du répertoire `bin`. Pour plus de détails sur le démarrage d'un service de catalogue, voir «Démarrage du service de catalogue dans un environnement autonome», à la page 331.

Trace sur un serveur de conteneur autonome

Vous pouvez activer la trace sur un serveur de conteneur de deux manières. Vous pouvez créer un fichier de propriétés de serveur comme expliqué dans la section des journaux ou activer la trace à l'aide de la ligne de commande, au démarrage. Pour activer la trace du conteneur avec un fichier de propriétés de serveur, mettez à jour la propriété **traceSpec** avec la spécification de trace requise. Pour activer la trace du conteneur à l'aide de paramètres de démarrage, utilisez les paramètres **-traceSpec** et **-traceFile**. Par exemple :

```
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 vous démarrez le serveur dans le répertoire *<rép_base_eXtremeScale>*`/bin`, les journaux et les fichiers de trace se trouvent dans les répertoires `logs/`*<nom_serveur>* du répertoire `bin` Pour plus de détails sur le démarrage d'un processus de conteneur, voir la rubrique «Démarrage de processus de conteneur», à la page 333.

Trace avec l'interface ObjectGridManager

Une autre option consiste à définir la trace lors de la phase d'exécution sur une interface ObjectGridManager. La définition de la sur une interface ObjectGridManager permet d'extraire la trace sur un client eXtreme Scale lorsqu'il se connecte à eXtreme Scale et valide des transactions. Pour définir la trace sur une interface ObjectGridManager, fournissez une spécification de trace et un journal de trace.

```
ObjectGridManager manager = ObjectGridManagerFactory.getObjectGridManager();
...
manager.setTraceEnabled(true);
manager.setTraceFileName("logs/myClient.log");
manager.setTraceSpecification("ObjectGridReplication=all=enabled");
```

Activation de la trace à l'aide de l'utilitaire xsadmin

Pour activer la trace à l'aide de l'utilitaire xsadmin, utilisez l'option **setTraceSpec**. Utilisez l'utilitaire xsadmin pour activer la trace sur un environnement autonome lors de la phase d'exécution et non au démarrage. Vous pouvez activer la trace sur tous les serveurs et services de catalogue ou filtrer les serveurs en fonction du nom ObjectGrid, etc. Par exemple, pour activer la trace ObjectGridReplication avec accès au serveur du service de catalogue, exécutez :

```
<base_eXtremeScale>/bin>xsadmin.bat -setTraceSpec "ObjectGridReplication=all=enabled"
```

Vous pouvez également désactiver la trace en affectant à la spécification de trace la valeur `*=all=disabled`.

Pour plus d'informations, voir «Surveillance à l'aide de l'exemple d'utilitaire xsAdmin», à la page 387.

Fichiers et répertoires ffdc

Les fichiers FFDC sont destinés au support technique d'IBM, pour le débogage. Ces fichiers peuvent être demandés par le support technique d'IBM, en cas de problème.

Ces fichiers se trouvent dans un répertoire libellé `ffdc` et contiennent des fichiers similaires au suivant :

```
server2_exception.log
server2_20802080_07.03.05_10.52.18_0.txt
```

Options de trace

Vous pouvez activer la trace pour fournir des informations sur votre environnement au service d'assistance IBM.

A propos de la trace

La trace de WebSphere eXtreme Scale est divisée en plusieurs composants. Comme pour la trace WebSphere Application Server, vous pouvez spécifier le niveau de trace à utiliser. Les niveaux de trace courants sont les suivants : all, debug, entryExit et event.

Voici un exemple de chaîne de trace :

```
ObjectGridComponent=level=enabled
```

Vous pouvez concaténer les chaînes de trace. Utilisez le symbole `*` (astérisque) pour spécifier une valeur générique, telle que `ObjectGrid*=all=enabled`. Si vous devez fournir une trace au service d'assistance IBM, une chaîne de trace spécifique est demandée. Par exemple, en cas de problème de réplication, la trace `ObjectGridReplication=debug=enabled` peut être demandée.

Spécification de la trace

ObjectGrid
Moteur général du cache central.

ObjectGridCatalogServer
Service de catalogue général.

ObjectGridChannel
Communications statiques de la topologie de déploiement.

7.1+ **ObjectGridClientInfo**
Informations sur le client DB2.

7.1+ **ObjectGridClientInfoUser**
Informations sur l'utilisateur DB2.

ObjectgridCORBA
Communications dynamiques de la topologie de déploiement.

ObjectGridDataGrid
API AgentManager.

ObjectGridDynaCache
Fournisseur de cache dynamique de WebSphere eXtreme Scale.

ObjectGridEntityManager
API EntityManager. A utiliser avec l'option Projector.

ObjectGridEvictors
Expulseurs pré-intégrés d'ObjectGrid.

ObjectGridJPA
Chargeurs JPA (Java Persistence API).

ObjectGridJPACache
Plug-in de cache JPA.

ObjectGridLocking
Gestionnaire de verrouillage des entrées de cache d'ObjectGrid.

ObjectGridMBean
Beans de gestion.

7.1+ **ObjectGridMonitor**
Infrastructure de la surveillance de l'historique.

ObjectGridPlacement
Service de positionnement des fragments de serveur de catalogues.

ObjectGridQuery
Requête ObjectGrid.

ObjectGridReplication
Service de réplication.

ObjectGridRouting
Détails du routage client/serveur.

ObjectGridSecurity
Trace de la sécurité.

ObjectGridStats
Statistiques d'ObjectGrid.

ObjectGridStreamQuery
API de la requête de flux.

ObjectGridWriteBehind
Ecriture différée d'ObjectGrid.

Projector
Moteur dans l'API EntityManager.

QueryEngine
Moteur de requête des API de requête Object et EntityManager.

QueryEnginePlan
Diagnostiques du plan de requête.

IBM Support Assistant for WebSphere eXtreme Scale

IBM Support Assistant permet de collecter des données, d'analyser des symptômes et d'accéder à des informations sur les produits.

IBM Support Assistant Lite

IBM Support Assistant Lite for WebSphere eXtreme Scale assure une collecte automatique des données et l'analyse des symptômes pour l'identification des problèmes et de leurs causes.

IBM Support Assistant Lite réduit le temps consacré à la reproduction des problèmes en adaptant son ensemble de niveaux de traçabilité (niveaux de fiabilité, de disponibilité et de facilité de maintenance, qui sont définis automatiquement par l'outil) afin de simplifier l'identification des problèmes. Mais si cette assistance ne suffit pas et que vous ayez besoin de l'aide d'un technicien, IBM Support Assistant Lite réduit également le temps consacré à l'envoi des informations appropriées au support technique d'IBM.

IBM Support Assistant Lite est inclus dans chaque installation de WebSphere eXtreme Scale version 7.1.0

IBM Support Assistant

IBM® Support Assistant (ISA) permet d'accéder rapidement à des ressources de produits, de formation et de support qui pourront vous aider à répondre de vous-mêmes à vos questions et à résoudre les problèmes rencontrés avec des logiciels IBM sans avoir besoin de contacter le support IBM. Différents plug-in spécifiques à différents produits vous permettent de personnaliser IBM Support Assistant en fonction des produits particuliers que vous avez installés. IBM Support Assistant peut également collecter des données système, des fichiers journaux et d'autres informations qui aideront le support technique d'IBM à déterminer la cause des problèmes.

IBM Support Assistant est un utilitaire qui s'installe sur le poste de travail et non sur le serveur WebSphere eXtreme Scale lui-même. En effet, sa mémoire et ses besoins en ressources risqueraient d'affecter de manière négative les performances du serveur WebSphere eXtreme Scale. Les composants portables de diagnostics qui sont inclus dans l'Assistant sont conçus pour avoir un impact minimal sur le fonctionnement normal d'un serveur.

IBM Support Assistant est utilisable des manières suivantes :

- pour effectuer des recherches dans des sources IBM et non IBM de connaissances et d'information sur plusieurs produits IBM afin de répondre à une question ou de résoudre un problème
- pour trouver des informations complémentaires dans des ressources Web dédiées à un produit donné (pages d'accueil du produit et de son support,

groupes de discussions et forums d'utilisateurs, ressources d'acquisition de compétences et de formations, informations de résolution des problèmes et FAQ)
- pour renforcer vos capacités à diagnostiquer les problèmes d'un produit donné grâce aux outils de diagnostics ciblés proposés par l'Assistant
- pour simplifier la collecte des données de diagnostic afin de vous aider, IBM et vous, à résoudre vos problèmes (collecte de données générales ou liées à un symptôme particulier)
- pour vous aider à signaler des problèmes au support IBM via une interface personnalisée en ligne avec possibilité d'attacher aux incidents signalés les données de diagnostic mentionnées plus haut ou toute autre information

Enfin, la fonctionnalité Updater intégrée permet de mettre à jour le support pour d'autres produits logiciels et d'autres fonctionnalités au fur et à mesure de leur disponibilité. Pour configurer IBM Support Assistant afin de l'utiliser avec WebSphere eXtreme Scale, commencez par l'installer à l'aide des fichiers fournis dans l'image téléchargée à partir de la page Web IBM Support Overview (http://www-947.ibm.com/support/entry/portal/Overview/Software/Other_Software/IBM_Support_Assistant. Ensuite, utilisez IBM Support Assistant pour repérer et installer les mises à jour de produits qui vous intéressent. Vous pouvez également choisir d'installer des plug-in pour d'autres logiciels IBM de votre environnement. Vous trouverez des informations complémentaires et la dernière version d'IBM Support Assistant à la page Web IBM Support Assistant (http://www.ibm.com/software/support/isa/).

Messages

Lorsqu'un message apparaît dans le journal ou d'autres parties de l'interface du produit, vous pouvez le rechercher en fonction de son préfixe de composant pour obtenir de plus amples informations.

Recherche de messages

Lorsque vous rencontrez un message dans un journal, copiez le numéro du message avec sa lettre préfixe et son numéro et effectuez une recherche dans le Centre de documentation (par exemple, `CWOBJ1526I`). Lorsque vous recherchez le message, vous pouvez trouver une explication supplémentaire du message et les éventuelles actions à effectuer pour résoudre le problème.

Pour l'index des messages du produit, reportez-vous au Centre de documentation.

Notes sur l'édition

Des liens permettent d'accéder au site Web de support technique, à la documentation, aux dernières mises à jour, aux restrictions et aux incidents recensés du produit.

- «Dernières mises à jour, limitations et problèmes connus», à la page 455
- «Accès à la configuration système et logicielle requise», à la page 455
- «Accès à la documentation du produit», à la page 455
- «Accès au site de support technique du produit», à la page 455
- «Contacter le service de support logiciel IBM», à la page 455

Dernières mises à jour, limitations et problèmes connus

Les notes sur l'édition sont disponibles sur le site de support technique du produit sous forme de notes techniques. Pour afficher une liste de toutes les notes techniques de WebSphere eXtreme Scale, accédez à la page Web du support technique. Cliquer sur les liens indiqués ici générera une recherche dans la page Web du support des notes sur l'édition correspondantes, lesquelles seront retournées sous forme de liste.

- 7.1+ Pour voir la liste des notes sur l'édition de la version 7.1, allez à la page Web du support.
- Pour voir la liste des notes sur l'édition de la version 7.0, allez à la page Web du support.
- Pour voir la liste des notes sur l'édition de la version 6.1, allez au wiki des notes sur l'édition.

Accès à la configuration système et logicielle requise

La configuration matérielle et logicielle requise est détaillée dans les pages suivantes :

- Configuration requise détaillée

Accès à la documentation du produit

Pour accéder à l'ensemble des informations, accédez à la page Bibliothèque.

Accès au site de support technique du produit

Pour rechercher les informations de support technique et notamment les dernières notes techniques, les fichiers à télécharger et les correctifs, accédez à la page du support technique.

Contacter le service de support logiciel IBM

Si un incident survient lors de l'utilisation du produit, essayez tout d'abord d'effectuer les opérations suivantes :

- Suivez les étapes décrites dans la documentation du produit
- Recherchez la documentation connexe dans l'aide en ligne
- Recherchez les messages d'erreur dans le document de référence des messages

Si vous ne parvenez pas à résoudre l'erreur à l'aide d'une des méthodes précédentes, prenez contact avec le support technique IBM.

Remarques

Les références aux produits, logiciels et services d'IBM n'impliquent pas qu'ils soient distribués dans tous les pays dans lesquels IBM exerce son activité. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. L'évaluation et la vérification de son fonctionnement en conjonction avec d'autres produits, hormis ceux expressément désignés par IBM, relèvent de la responsabilité de l'utilisateur.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, New York 10594 USA

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Marques

Les termes suivant sont des marques d'IBM Corporation aux Etats-Unis et/ou dans certains autres pays :

- AIX
- CICS
- Cloudscape
- DB2
- Domino
- IBM
- Lotus
- RACF
- Redbooks
- Tivoli
- WebSphere
- z/OS

Java ainsi que tous les logos et toutes les marques incluant Java sont des marques de Sun Microsystems, Inc. aux Etats-Unis et/ou dans certains autres pays.

LINUX est une marque de Linus Torvalds aux Etats-Unis et/ou dans certains autres pays.

Microsoft, Windows, Windows NT® et le logo Windows sont des marques de Microsoft Corporation aux Etats-Unis et/ou dans certains autres pays.

UNIX est une marque enregistrée de l'Open Group aux Etats-Unis et dans d'autres pays.

Les autres noms de sociétés, de produits et de services peuvent appartenir à des tiers.

Index

L

M

N

O

P

Q

R

S

T

V

W

X

Z