WebSphere eXtreme Scale Version 7.0
Administratorhandbuch

WebSphere eXtreme Scale Administratorhandbuch

WebSphere® eXtreme Scale Version 7.0
Administratorhandbuch

WebSphere eXtreme Scale Administratorhandbuch

Diese Edition bezieht sich auf Version 7, Release 0 von WebSphere eXtreme Scale und alle nachfolgenden Releases und Bearbeitungen, sofern kein anderer Hinweis in den neuen Editionen enthalten ist.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellen

Informationen zu *Administratorhandbuch*

Das Dokumentationsset von WebSphere eXtreme Scale umfasst drei Handbücher, die die erforderlichen Informationen zur Verwendung des Produkts WebSphere eXtreme Scale, die Programmierung für das Produkt und die Verwaltung des Produkts enthalten.

Bibliothek von WebSphere eXtreme Scale

Die Bibliothek von WebSphere eXtreme Scale enthält die folgenden Bücher:

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

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

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

Zielgruppe

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

Aufbau des Handbuchs

Das Buch enthält Informationen zu den folgenden wichtigen Themen:

- **Kapitel 1** enthält einführende Informationen.
- **Kapitel 2** enthält Informationen zur Planung der Anwendungsimplementierung.
- **Kapitel 3** enthält Informationen zur Kapazitätsplanung.
- **Kapitel 4** enthält Informationen zum Installieren des Produkt.
- **Kapitel 5** enthält Informationen zur Anpassung für die Plattform z/OS.
- **Kapitel 6** enthält Informationen zum Konfigurieren des Produkts.
- **Kapitel 7** enthält Informationen zur Verwaltung der Umgebung.
- **Kapitel 8** enthält Informationen zur Überwachung Ihrer Implementierungsumgebung.
- **Kapitel 9** enthält Informationen zum Sichern der Implementierungsumgebung.
- **Kapitel 10** enthält Informationen zur Fehlerbehebung in der Umgebung.
- **Kapitel 11** enthält das Produktglossar.

Aktualisierungen für dieses Handbuch

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

Hinweise zu Rückmeldungen

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

Kapitel 1. Einführung in WebSphere eXtreme Scale

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

Die eigenständige Installation von WebSphere eXtreme Scale enthält ein Beispiel, das Sie verwenden können, um Ihre Installation zu prüfen und eigene Erfahrungen mit der Verwendung eines einfachen eXtreme-Scale-Grids und -Clients sammeln können. Das Einführungsbeispiel ist im Verzeichnis *Installationsstammverzeichnis*/ObjectGrid/gettingstarted enthalten.

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

Scripts und ihre Funktionen

Dieses Beispiel stellt die folgenden vier Scripts bereit:

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

- ./env.sh
- env.bat

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

- ./runcat.sh
- runcat.bat

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

- ./runcontainer.sh *eindeutiger_Servername*
- runcontainer.bat *eindeutiger_Servername*

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

- ./runclient.sh *Befehl Wert1 Wert2*
- runclient.sh *Befehl Wert1 Wert2*

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

- Geben Sie i ein, um *Wert2* in das Grid mit dem Schlüssel *Wert1* einzufügen.
- Geben Sie u ein, um das Objekt mit dem Schlüssel *Wert1* in *Wert2* zu aktualisieren.

- Geben Sie `d` ein, um das Objekt mit dem Schlüssel *Wert1* zu löschen.
- Geben Sie `g` ein, um das Objekt mit dem Schlüssel *value1* abzurufen und anzuzeigen.

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

Grundlegende Schritte

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

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

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

   Ersetzen Sie *Installationsstammverzeichnis* durch den Pfad des Installationsstammverzeichnisses von eXtreme Scale bzw. durch den Stammdateipfad des *Installationsstammverzeichnisses* der extrahierten Testversion von eXtreme Scale.
3. Setzen oder exportieren Sie die Umgebungsvariable JAVA_HOME so, dass sie auf das Installationsverzeichnis eines gültigen JDK bzw. einer gültigen JRE Version 1.5 oder höher verweist.

   export JAVA_HOME=*Java-Ausgangsverzeichnis*

   set JAVA_HOME=*Java-Ausgangsverzeichnis*
4. Führen Sie das folgende Script aus, um einen Katalogserviceprozess auf localhost zu starten:
   - `./runcat.sh`
   - `runcat.bat`

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

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

- Löschen Sie den Wert:
  - `./runclient.sh d key1`
  - `runclient.bat d key1`

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

ObjectGrid definieren

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

ObjectGrid-XML-Deskriptordatei

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

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

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

</objectGridConfig>
```

XML-Implementierungsrichtliniendeskriptordatei

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

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

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

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

</deploymentPolicy>
```

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

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

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

ObjectGrid verwenden

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

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

1. Durch den Abruf einer ClientClusterContext-Instanz eine Verbindung zum Katalogservice herstellen,
2. Abruf einer ObjectGrid-Clientinstanz,
3. Abruf einer Session-Instanz,
4. Abruf einer ObjectMap-Instanz,
5. Verwendung der ObjectMap-Methoden.

1. Durch den Abruf einer ClientClusterContext-Instanz eine Verbindung zum Katalogservice herstellen

Zum Herstellen einer Verbindung zum Katalogserver können Sie die Methode "connect" der API "ObjectGridManager" verwenden. Die Methode "connect", die

von diesem Beispiel verwendet wird, erfordert nur den Katalogserverendpunkt im Format `Hostname:Port`, z. B. `localhost:2809`. Wenn die Verbindungsherstellung zum Katalogserver erfolgreich ist, gibt die Methode ″connect″ eine ClientClusterContext-Instanz zurück. Die ClientClusterContext-Instanz ist erforderlich, um das ObjectGrid über die API ″ObjectGridManager″ abzurufen. Das folgende Code-Snippet demonstriert, wie die Verbindung zu einem Katalogserver hergestellt und eine ClientClusterContext-Instanz abgerufen wird.

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

2. Abruf einer ObjectGrid-Instanz

Zum Abrufen einer ObjectGrid-Instanz können Sie die Methode ″getObjectGrid″ der API ″ObjectGridManager″ verwenden. Die Methode ″getObjectGrid″ erfordert die ClientClusterContext-Instanz und den Namen der ObjectGrid-Instanz. Die ClientClusterContext-Instanz wird während der Verbindungsherstellung zum Katalogserver abgerufen. Der Name des ObjectGrids ist ″Grid″ und wurde in der Datei `objectgrid.xml` angegeben. Das folgende Code-Snippet demonstriert, wie Sie ein ObjectGrid durch Aufruf der Methode ″getObjectGrid″ der API ″ObjectGridManager″ abrufen können:

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

3. Abruf einer Session-Instanz

Sie können eine Session-Instanz über die abgerufene ObjectGrid-Instanz abrufen. Eine Session-Instanz ist erforderlich, um die ObjectMap-Instanz abzurufen und die Transaktionsabgrenzung durchzuführen. Das folgende Code-Snippet demonstriert, wie Sie eine Session-Instanz durch Aufruf der Methode ″getSession″ der API ″ObjectGrid″ abrufen:

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

4. Abruf einer ObjectMap-Instanz

Nach dem Abruf einer Session-Instanz können Sie über diese eine ObjectMap-Instanz abrufen, indem Sie die Methode ″getMap″ der API ″Session″ aufrufen. Sie müssen den Namen der Map als Parameter an die Methode ″getMap″ übergeben, um die ObjectMap-Instanz abzurufen. Das folgende Code-Snippet demonstriert, wie Sie eine ObjectMap-Instanz durch den Aufruf der Methode ″getMap″ der API ″Session″ abrufen:

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

5. Verwendung der ObjectMap-Methoden

Nach dem Abruf einer ObjectMap-Instanz können Sie sie über die API ″ObjectMap″ starten. Beachten Sie, dass die ObjectMap eine Transaktions-Map ist und eine Transaktionsabgrenzung durch die Verwendung der Methoden ″begin″ und ″commit″ der API ″Session″ erfordert. Wenn es keine explizite Transaktionsabgrenzung gibt, werden ObjectMap-Operationen mit Transaktionen mit automatischem Festschreiben ausgeführt.

Das folgende Code-Snippet demonstriert, wie Sie die API ″ObjectMap″ mit Transaktionen mit automatischem Festschreiben verwenden:

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

Das folgende Code-Snippet demonstriert, wie Sie die API ″ObjectMap″ mit expliziter Transaktionsabgrenzung verwenden:

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

Weitere Informationen

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

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

Kapitel 2. Anwendungsimplementierung planen

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

Implementierungskonfigurationen für eXtreme Scale

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

Lokale Umgebungen

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

Verteilte Umgebungen

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

Hardware- und Softwarevoraussetzungen

Für die Verwendung von WebSphere eXtreme Scale wird keine bestimmte Hardware- oder Betriebssystemversion vorausgesetzt. Die offiziellen Unterstützungserklärungen für eXtreme Scale finden Sie auf der Webseite zu den Systemvoraussetzungen.

Eine detaillierte Aufstellung der unterstützten Hardware- und Softwareoptionen nach Betriebssystem für WebSphere eXtreme Scale finden Sie auf der Webseite mit den Systemvoraussetzungen.

Sollten die im Information Center enthaltenen Informationen und die Informationen auf der Webseite zu den Systemvoraussetzungen widersprüchlich sein, haben die Informationen auf der Website Vorrang. Die Informationen zu den Voraussetzungen im Information Center werden nur aus Komfortgründen bereitgestellt.

Hardwarevoraussetzungen

WebSphere eXtreme Scale setzt keine bestimmte Hardwareversion voraus. Die Hardwarevoraussetzungen richten sich nach der unterstützten Hardware für die

Installation von Java Platform, Standard Edition, die Sie für die Ausführung von WebSphere eXtreme Scale verwenden. Wenn Sie eXtreme Scale mit WebSphere Application Server oder einer anderen Java-EE-Implementierung (Java Platform, Enterprise Edition) verwenden, sind die Hardwarevoraussetzungen dieser Plattformen für WebSphere eXtreme Scale ausreichend.

Betriebssystemvoraussetzungen

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

Voraussetzungen in Bezug auf WebSphere Application Server

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

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

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

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

Weitere Voraussetzungen in Bezug auf den Anwendungsserver

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

Leistung optimieren

Zur Verbesserung der Leistung können Sie verschiedene Punkte in Betracht ziehen, wie z. B. die Optimierung des Betriebssystems und des Netzes, die Planung von Netzanschlüssen, die Optimierung der Java Virtual Machine (JVM) für die Einstellungen von WebSphere eXtreme Scale, die Konfiguration von Failover und weitere Optimierungsmaßnahmen.

Netzports planen

WebSphere eXtreme Scale ist ein verteilter Cache, der das Öffnen von Ports für die Kommunikation mit dem ORB (Object Request Broker) und dem TCP-Stack (Transmission Control Protocol) zwischen Java Virtual Machines und anderen Maschinen voraussetzt. Sie müssen Ihre Ports planen und steuern, insbesondere in einer Firewallumgebung, z. B., wenn Sie Katalogservices und Container zum Öffnen mehrerer Ports verwenden.

Katalogservice-Grid

Ein Katalogservice-Grid setzt Folgendes voraus:

1. einen Peer-Port für den High-Availability Manager (oder kurz HA-Manager), über den die Kommunikation zwischen Peer-Katalogservern über einen TCP-Stack erfolgt,
2. einen Clientport, über den die Katalogserver auf Daten des Katalogservice zugreifen,
3. JMXServicePort, um den Port festzulegen, den der JMX-Service verwenden soll,
4. einen ORB-Listener-Port für Container und Clients, über den die Kommunikation mit dem Katalogservice über den ORB erfolgt.

Die zuvor aufgelisteten Ports können Sie, wie im folgenden Beispiel gezeigt, mit dem Befehl startOgServer für eine eigenständige Implementierung angeben:

```
–catalogServiceEndPoints cs1:host1:clientPort:peerPort, cs2:host2:clientPort:peerPort,
cs3:host3:clientPort:peerPort -listenerPort <ORB-Port> -JMXServicePort <JMX-Port>
```

In einer Umgebung mit WebSphere Application Server werden die in der vorherigen Liste aufgeführten Ports mit angepassten Eigenschaften mit dem Schlüssel ″catalog.services.cluster″ für die WebSphere-Zelle wie folgt angegeben:

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

Die Containerserver von WebSphere eXtreme Scale erfordern für den Betrieb ebenfalls verschiedene Ports. Standardmäßig generiert der Containerserver von eXtreme Scale seinen HA-Manager-Port und seinen ORB-Listener-Port automatisch mit dynamischen Ports. Da es in der Firewallumgebung empfehlenswert ist, Ports zu planen und zu steuern, werden Optionen bereitgestellt, um Containerserver von eXtreme Scale, wie im folgenden Beispiel gezeigt, mit dem angegebenen HA-Manager-Port und dem angegebenen ORB-Listener-Port mit einer Option im Befehl ″startOgServer″ zu starten:

```
-HaManagerPort <Peer-Port>
-listenerPort <ORB-Port>
```

Eine ordnungsgemäße Planung der Portsteuerung ist vorteilhaft, aber die Planung und Verwaltung dieser Ports kann sich natürlich schwierig gestalten, wenn Hunderte von Java Virtual Machines in einer Maschine gestartet werden. Jeder Portkonflikt führt zu einem Fehler beim Serverstart.

Wenn die Sicherheit aktiviert ist, muss der vorherigen Portliste ein SSL-Port (Secure Sockets Layer) hinzugefügt werden. Mit `Dcom.ibm.CSI.SSLPort=<sslPort>` als Argument für ″-jvmArgs″ kann der gewünschte `<SSL-Port>` festgelegt werden.

Optimierung von Betriebssystem und Netz

Eine Netzoptimierung kann durch Änderung der Keepalive-Einstellungen für Verbindungen Verzögerungen im TCP-Stack (Transmission Control Protocol) reduzieren und den Durchsatz durch Änderung der TCP-Puffer verbessern.

Betriebssysteme

Ein Windows®-System erfordert die wenigste Optimierung, ein Solaris-System die meiste. Die folgenden Informationen gelten für jedes angegebene System und können die Leistung von WebSphere eXtreme Scale verbessern. Sie müssen die Optimierung Ihrem Netz und Ihrer Anwendungslast entsprechend optimieren.

Windows

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\
Tcpip\Parameters
MaxFreeTcbs = dword:00011940
MaxHashTableSize = dword:00010000
MaxUserPort = dword:0000fffe
TcpTimedWaitDelay = dword:0000001e
```

Solaris

```
ndd -set /dev/tcp tcp_time_wait_interval 60000
fndd -set /dev/tcp tcp_keepalive_interval 15000
ndd -set /dev/tcp tcp_fin_wait_2_flush_interval 67500
ndd -set /dev/tcp tcp_conn_req_max_q 16384
ndd -set /dev/tcp tcp_conn_req_max_q0 16384
ndd -set /dev/tcp tcp_xmit_hiwat 400000
ndd -set /dev/tcp tcp_recv_hiwat 400000
ndd -set /dev/tcp tcp_cwnd_max 2097152
ndd -set /dev/tcp tcp_ip_abort_interval 20000
ndd -set /dev/tcp tcp_rexmit_interval_initial 4000
ndd -set /dev/tcp tcp_rexmit_interval_max 10000
ndd -set /dev/tcp tcp_rexmit_interval_min 3000
ndd -set /dev/tcp tcp_max_buf 4194304
```

AIX

```
/usr/sbin/no -o tcp_sendspace=65536
/usr/sbin/no -o tcp_recvspace=65536
/usr/sbin/no -o udp_sendspace=65536
/usr/sbin/no -o udp_recvspace=65536
/usr/sbin/no -o somaxconn=10000
/usr/sbin/no -o tcp_nodelayack=1
/usr/sbin/no –o tcp_keepinit=40
/usr/sbin/no –o tcp_keepintvl=10
```

LINUX

```
sysctl -w net.ipv4.tcp_timestamps=0
sysctl -w net.ipv4.tcp_tw_reuse=1
sysctl -w net.ipv4.tcp_tw_recycle=1
sysctl -w net.ipv4.tcp_fin_timeout=30
sysctl -w net.ipv4.tcp_keepalive_time=1800
sysctl -w net.ipv4.tcp_rmem="4096 87380 8388608"
sysctl -w net.ipv4.tcp_wmem="4096 87380 8388608"
sysctl -w net.ipv4.tcp_max_syn_backlog=4096
```

HP-UX

```
ndd -set /dev/tcp tcp_ip_abort_cinterval 20000
```

ORB-Eigenschaften und Dateideskriptoreinstellungen

Die Optimierungshinweise beziehen sich unter anderem auf die ORB-Eigenschaften (Object Request Broker) und die Dateideskriptoreinstellungen.

ORB-Eigenschaften

Der ORB (Object Request Broker) wird von WebSphere eXtreme Scale für die Kommunikation über einen TCP-Stack verwendet. Sie finden die erforderliche Datei `orb.properties` im Verzeichnis `java/jre/lib`. Für Lastspitzen mit großen Objekten aktivieren Sie die ORB-Fragmentierung mit der folgenden Einstellung:

```
com.ibm.CORBA.FragmentSize=<richtige Größe>
```

Mit der folgenden Einstellung können Sie das Anwachsen des Thread-Pools verhindern:

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

Mit den folgenden Einstellungen können Sie angemessene Zeitlimits festlegen, um die Ausführung zu vieler Threads in einer abnormalen Situation zu verhindern:

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

Dateideskriptor

Auf UNIX®- und Linux®-Systemen ist die zulässige Anzahl offner Dateien pro Prozess beschränkt. Das Betriebssystem gibt die zulässige Anzahl offener Dateien an. Wenn dieser Wert zu klein ist, tritt ein Fehler bei der Speicherzuordnung unter AIX auf, und es wird protokolliert, dass zu viele Dateien offen sind.

Setzen Sie diese Einstellung im Terminalfenster auf dem UNIX-System auf einen höheren Wert als den Systemstandardwert. Für große SMP-Maschinen mit Klonen legen Sie den Wert für eine uneingeschränkte Anzahl offener Dateien fest.

Für AIX-Konfiguration setzen Sie diese Einstellung mit dem Befehl `ulimit -n -1` auf den Wert `-1` (uneingeschränkt).

Für Solaris-Konfigurationen setzen Sie diese Einstellung mit dem Befehl `ulimit -n 16384` auf den Wert `16384`.

Zum Anzeigen des aktuellen Werts verwenden Sie den Befehl `ulimit –a`.

JVM-Optimierung für WebSphere eXtreme Scale

In diesem Abschnitt werden einige spezielle Fragen zur Optimierung von Java Virtual Machines (JVM) für WebSphere eXtreme Scale beantwortet. Wenn Ihr System Leistungsprobleme aufweist, wenden Sie sich an die IBM Unterstützungsfunktion. Diese Seite wird mit weiteren Informationen zur Optimierung aktualisiert, sobald diese verfügbar sind.

In den meisten Fällen erfordert WebSphere eXtreme Scale nur wenige bzw. keine speziellen JVM-Einstellungen. Wenn Sie sehr viele Objekte haben, die in WebSphere eXtreme Scale gespeichert werden, passen Sie die Größe des Heap-Speichers entsprechend an, um Speicherengpässe zu vermeiden.

Plattformen

Leistungstests wurden hauptsächlich auf AIX- (32 Wege), Linux- (4 Wege), Windows (8 Wege) und AZUL-Maschinen (384 Wege) durchgeführt. Auf AZUL-Systemen und High-End-AIX-Maschinen konnten Multithread-Szenarios mit hoher Last realisiert und Konfliktpunkte identifiziert und ausgeräumt werden. Außerdem wurde auf AZUL-Systemen die Garbage-Collection weggelassen, die die Skalierungsläufe während der Tests erheblich verfälschen kann.

ORB-Anforderungen (Object Request Broker)

IBM® SDK enthält eine IBM ORB-Implementierung, die mit WebSphere Application Server und WebSphere eXtreme Scale getestet wurde. Zur Vereinfachung des Unterstützungsprozesses sollten Sie eine von IBM bereitgestellte JVM verwenden. Andere JVM-Implementierungen verwenden einen anderen ORB. Der IBM ORB wird nur unverändert mit den von JVMs von IBM bereitgestellt. WebSphere eXtreme Scale erfordert für den Betrieb einen funktionierenden ORB. Sie können WebSphere eXtreme Scale zwar auch mit ORBs anderer Anbieter verwenden, aber wenn ein Problem im ORB auftritt, müssen Sie sich um Unterstützung an den ORB-Anbieter wenden. Die IBM ORB-Implementierung ist mit JVMs anderer Anbieter kompatibel und kann bei Bedarf ersetzt werden.

Garbage-Collection

WebSphere eXtreme Scale erstellt temporäre Objekte für jede Transaktion, wie z. B. Anforderungs- und Antwortobjekte und eine Protokollfolge. Da sich diese Objekte auf die Effizienz der Garbage-Collection auswirken, ist die Optimierung der Garbage-Collection kritisch.

Verwenden Sie für die IBM Virtual Machine for Java den Collector ″optavgpause″ in Szenarios mit einer hohen Aktualisierungsrate (100 % geänderter Transaktionseinträge). Der Collector ″gencon″ funktioniert in Szenarios, in denen die Daten nur relativ selten aktualisiert werden (10 % der Zeit oder weniger), viel besser als der Collector ″optavgpause″. Experimentieren Sie mit beiden Collector, um festzustellen, welcher sich besser in Ihrem Szenario eignet. Wenn Sie ein Leistungsproblem erkennen, aktivieren Sie die ausführliche Garbage-Collection, um die für die Garbage-Collection aufgebrachte Zeit in Prozent zu überprüfen. Es wurden Szenarios gefunden, in denen 80 % der Zeit für die Garbage-Collection aufgebracht wurde, bis das Problem durch Optimierung behoben wurde.

Weitere Informationen zum Konfigurieren der Garbage-Collection finden Sie unter IBM Virtual Machine for Java optimieren.

JVM-Leistung

WebSphere eXtreme Scale kann unter verschiedenen Versionen von Java 2 Platform, Standard Edition (J2SE) ausgeführt werden. WebSphere eXtreme Scale Version 6.1 unterstützt J2SE Version 1.4.2 und höher. Zur Steigerung der Entwicklerproduktivität und der Leistung verwenden Sie J2SE 5 oder höher, um die Vorteile von Annotationen und verbesserter Garbage-Collection zu nutzen. WebSphere eXtreme Scale kann in 32-Bit- und 64-Bit-JVMs ausgeführt werden.

WebSphere eXtreme Scale wurde mit einem Teil der verfügbaren virtuellen Maschinen getestet, aber die Liste der unterstützten JVMs ist nicht exklusiv. Sie können WebSphere eXtreme Scale in jeder Version 1.4.2 oder höher ausführen, aber wenn ein Problem in der JVM auftritt, müssen Sie sich für Unterstützung an den jeweiligen JVM-Anbieter wenden. Verwenden Sie, sofern möglich, die JVM aus der WebSphere-Laufzeitumgebung auf jeder Plattform, die von WebSphere Application Server unterstützt wird.

In den meisten Szenarios, in denen WebSphere eXtreme Scale verwendet wird, bietet Java Platform Standard Edition 6 der JVM eine bessere Leistung als Edition 5 oder 1.4. Die Leistung von Java 2 Platform, Standard Edition Version 1.4 ist insbesondere in Szenarios, in denen der Collector ″gencon″ verwendet wird, sehr schwach. Java Platform Standard Edition 5 bietet eine gute Leistung, aber Java Platform, Standard Edition 6 eine bessere.

Große Heap-Speicher

Wenn die Anwendung ein großes Datenvolumen für jede Partition verwalten muss, kann die Garbage-Collection ein entscheidender Faktor sein. In einem Szenario, in dem hauptsächlich Leseoperationen durchgeführt werden, ist die Leistung selbst bei sehr großen Heap-Speichern (20 GB und mehr) gut, wenn eine Garbage-Collection nach Objektalter verwendet wird. Wenn der Heap-Speicher für die permanenten Objekte gefüllt ist, ist jedoch eine Pause zu bemerken, die proportional zur Größe des Live-Heap-Speichers und der Anzahl der Prozessoren in der Maschine ist. Diese Pause kann auf kleineren Maschinen mit großen Heap-Speichern sehr lang sein.

WebSphere eXtreme Scale unterstützt WebSphere Real Time Java. Mit WebSphere Real Time Java sind die Transaktionsverarbeitungsantworten für WebSphere eXtreme Scale konsistenter und vorhersehbar, und die Auswirkungen der Garbage-Collection und der Thread-Planung ist erheblich geringer. Die Auswirkungen werden so weit reduziert, dass die Standardabweichung bei den Antwortzeiten weniger als 10 % als beim regulären Java beträgt.

Weitere Informationen finden Sie im Abschnitt „WebSphere Real Time verwenden" auf Seite 222.

Thread-Anzahl

Die Thread-Anzahl ist von verschiedenen Faktoren abhängig. Die Anzahl der Threads, die von einem einzelnen Shard verwaltet werden können, ist begrenzt. Ein Shard ist eine Instanz einer Partition. Ein Shard kann ein primäres Shard oder ein Replikat-Shard sein. Je mehr Shards jede JVM enthält, desto mehr Threads und mehr gemeinsame Datenzugriffspfade sind möglich. Jedes Shard ist so nebenläufig wie möglich, aber nichtsdestotrotz gibt es einen Grenzwert.

Failover-Erkennung konfigurieren

Sie können das Intervall konfigurieren, in dem das System nach ausgefallenen Servern sucht. Dieser Wert wird als Intervall der Überwachungssignale bezeichnet.

Warum und wann dieser Vorgang ausgeführt wird

Die Konfiguration des Failovers variiert je nach Typ der verwendeten Umgebung. Wenn Sie eine eigenständige Umgebung verwenden, können Sie das Failover über die Befehlszeile konfigurieren. Wenn Sie eine Umgebung mit WebSphere Application Server Network Deployment verwenden, müssen Sie das Failover über die Administrationskonsole von WebSphere Application Server Network Deployment konfigurieren.

- Failover für eigenständige Umgebungen konfigurieren

  Sie können das Intervall der Überwachungssignale über die Befehlszeile mit dem Parameter **-heartbeat** in der Scriptdatei startOgServer.bat bzw. startOgServer.sh konfigurieren. Setzen Sie den Parameter auf einen der folgenden Werte:

Tabelle 1. Intervall der Überwachungssignale

Wert	Aktion	Beschreibung
0	Typisch (Standard-einstellung)	Failover werden gewöhnlich innerhalb von 30 Sekunden erkannt.
-1	Aggressiv	Failover werden gewöhnlich innerhalb von 5 Sekunden erkannt.
1	Gelockert	Failover werden gewöhnlich innerhalb von 180 Sekunden erkannt.

  Ein aggressives Intervall der Überwachungssignale kann hilfreich sein, wenn die Prozesse und das Netz stabil sind. Wenn das Netz oder die Prozesse nicht optimal konfiguriert sind, können Überwachungssignale verpasst werden, was zu einer falschen Fehlererkennung führen kann.

- Failover für Umgebungen mit WebSphere Application Server konfigurieren

  Sie können WebSphere Application Server Network Deployment Version 6.0.2 und höher so konfigurieren, dass ein schnelles Failover von WebSphere eXtreme Scale unterstützt wird. Die Standard-Failover-Zeit für permanente Fehler sind 200 Sekunden. Ein permanenter Fehler ist ein physischer Computer- oder Server-

absturz, das Ziehen des Netzkabels oder ein Betriebssystemfehler. Bei Fehlern aufgrund von Prozessabstürzen oder temporären Fehlern findet das Failover gewöhnlich in weniger als einer Sekunde statt. Die Fehlererkennung für temporäre Fehler findet statt, wenn die Netz-Sockets des inaktiven Prozesses für den Server, in dem der Prozess ausgeführt wird, automatisch vom Betriebssystem geschlossen werden.

Überwachungssignalkonfiguration für Stammgruppen

Wenn WebSphere eXtreme Scale in einem Prozess von WebSphere Application Server ausgeführt wird, werden die Failover-Merkmale aus den Stammgruppeneinstellungen des Anwendungsservers übernommen. In den folgenden Abschnitten wird beschrieben, wie Sie die Überwachungssignaleinstellungen der Stammgruppe für verschiedene Versionen von WebSphere Application Server Network Deployment konfigurieren:

– **Stammgruppeneinstellungen für WebSphere Application Server Network Deployment Version 6.x und 7.x aktualisieren**

  Geben Sie das Intervall der Überwachungssignale in WebSphere Application Server Version 6.0 bis Version 6.1.0.12 in Sekunden und ab Version 6.1.0.13 in Millisekunden an. Außerdem müssen Sie die Anzahl verpasster Überwachungssignale angeben. Dieser Wert gibt an, wie viele Überwachungssignale verpasst werden können, bevor eine Peer-JVM als ausgefallen betrachtet wird. Die Erkennungszeit für permanente Fehler entspricht in etwa dem Produkt aus Intervall der Überwachungssignale und Anzahl verpasster Überwachungssignale.

  Diese Eigenschaften werden mit Hilfe von angepassten Eigenschaften in der Stammgruppe über die WebSphere-Administrationskonsole angegeben. Einzelheiten zur Konfiguration finden Sie im Abschnitt Angepasste Eigenschaften der Stammgruppe. Diese Eigenschaften müssen für alle Stammgruppen angegeben werden, die von Anwendungen verwendet werden:

  - Das Intervall der Überwachungssignale wird mit der angepassten Eigenschaft IBM_CS_FD_PERIOD_SEC (in Sekunden) bzw. der angepassten Eigenschaft IBM_CS_FD_PERIOD_MILLIS (in Millisekunden) (erfordert Version 6.1.0.13 oder höher) angegeben.
  - Die Anzahl verpasster Überwachungssignale wird mit der angepassten Eigenschaft IBM_CS_FD_CONSECUTIVE_MISSED angegeben.

  Der Standardwert für die Eigenschaft IBM_CS_FD_PERIOD_SEC ist 20, und der Standardwert für die Eigenschaft IBM_CS_FD_CONSECUTIVE_MISSED ist 10. Wenn Sie die Eigenschaft IBM_CS_FD_PERIOD_MILLIS angeben, überschreibt diese jede definierte angepasste Eigenschaft IBM_CS_FD_PERIOD_SEC. Die Werte dieser Eigenschaften sind positive ganze Zahlen.

  Verwenden Sie die folgenden Einstellungen, um eine Erkennungszeit von 1500 ms für Server der WebSphere Application Server Network Deployment Version 6.x zu erzielen:

  - Setzen Sie `IBM_CS_FD_PERIOD_MILLIS = 750` (WebSphere Application Server Network Deployment Version 6.1.0.13 und höher).
  - Setzen Sie `IBM_CS_FD_CONSECUTIVE_MISSED = 2`.

– **Stammgruppeneinstellungen für WebSphere Application Server Network Deployment Version 7.0 aktualisieren**

  WebSphere Application Server Network Deployment Version 7.0 stellt zwei Stammgruppeneinstellungen bereit, die angepasst werden können, um die Failover-Erkennungszeit zu erhöhen oder zu verringern:

  - **Übertragungsintervall für Überwachungssignale.** Der Standardwert sind 30.000 Millisekunden.

- **Überwachungssignalzeitlimit.** Der Standardwert sind 180.000 Millisekunden.

Weitere Einzelheiten zum Ändern dieser Einstellungen finden Sie im WebSphere Application Server Network Deployment Information Center unter "Einstellungen für die Erkennung und Fehlererkennung".

Verwenden Sie die folgenden Einstellungen, um eine Fehlererkennungszeit von 1500 ms für Server der WebSphere Application Server Network Deployment Version 7 zu erzielen:

- Setzen Sie das Übertragungsintervall für Überwachungssignale auf 750 Millisekunden.
- Setzen Sie das Überwachungssignalzeitlimit auf 1500 Millisekunden.

Nächste Maßnahme

Wenn Sie diese Einstellungen ändern, um kürzere Failover-Zeiten anzugeben, müssen verschiedene Probleme bei der Systemoptimierung beachtet werden. Java ist keine Echtzeitumgebung. Es ist möglich, dass Threads verzögert werden, wenn die JVM lange Garbage-Collection-Zeiten verzeichnet. Threads können auch verzögert werden, wenn die Maschine, auf der die JVM ausgeführt wird, unter hoher Last steht (durch die JVM selbst oder durch andere Prozesse, die auf der Maschine ausgeführt werden). Wenn Threads verzögert werden, werden Überwachungssignale möglicherweise nicht rechtzeitig gesendet. Im schlimmsten Fall werden die durch die erforderliche Failover-Zeit verzögert. Wenn Threads verzögert werden, treten falsche Fehlererkennungen auf. Das System muss optimiert und dimensioniert werden, um sicherzustellen, dass falsche Fehlererkennungen in der Produktionsumgebung nicht auftreten. Dies kann am zuverlässigsten durch angemessene Lasttests sichergestellt werden.

Anmerkung: Die aktuelle Version von eXtreme Scale unterstützt WebSphere Real Time.

Katalogservice mit hoher Verfügbarkeit

Ein Katalogservice ist das Grid von Katalogservern, die Sie verwenden und die Topologieinformationen zu allen Containern in Ihrer Umgebung von eXtreme Scale bewahren. Der Katalogservice steuert den Lastausgleich und das Routing für alle Clients. Zum Implementieren von eXtreme Scale als speicherinternen Verarbeitungsbereich müssen Sie den Katalogservice für die hohe Verfügbarkeit als Cluster in ein Grid einfügen.

Komponenten des Katalogservice

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

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

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

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

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

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

Implementierung des Katalog-Grids

Stammgruppenmanager

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

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

Verteilungsservice

Der Katalogservice verwaltet die Verteilung von Shards auf die verfügbaren Container. Der Verteilungsservice ist dafür verantwortlich, dass die Ressourcen gleichmäßig auf die physischen Ressourcen verteilt werden. Der Verteilungsservice ordnet die einzelnen Shards ihren Hostcontainern zu. Er wird als ein über eine 1:N-Beziehung ausgewählter Service im Grid ausgeführt, so dass stets eine einzige Instanz des Service aktiv ist. Wenn diese Instanz ausfällt, wird ein anderer Prozess

ausgewählt, der die Arbeit dieser Instanz übernimmt. Aus Redundanzgründen wird der Status des Katalogservice in allen Servern, in denen der Katalogservice ausgeführt, repliziert.

Verwaltung

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

Positionsservice

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

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

Planung

Nach dem Starten eines Katalog-Grids werden die Member des Grids miteinander verbunden. Planen Sie die Topologie Ihres Katalog-Grids sorgfältig, weil die Katalogkonfiguration zur Laufzeit nicht geändert werden kann. Verteilen Sie das Grid so divers wie möglich, um Fehler zu verhindern.

Katalogserver-Grid starten

Container von eXtreme Scale, die in WebSphere Application Server integriert sind, zu einem eigenständigen Katalog-Grid verbinden

Sie können Container von eXtreme Scale, die in eine Umgebung von WebSphere Application Server integriert sind, zu einem eigenständigen Katalog-Grid verbinden. Verwenden Sie dieselbe Eigenschaft, um den Anwendungsserver mit dem Katalog-Grid zu verbinden. Die Eigenschaft verwaltet jedoch nicht den Lebenszyklus des Katalogs mit den Servern. Stattdessen ermöglicht die Eigenschaft den Containern, das ferne Katalog-Grid zu finden. Weitere Informationen finden Sie in .

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

Weitere Informationen finden Sie im Abschnitt mit den Informationen zu Katalogserver-Quorums im *Produktübersicht*.

Prüfliste für die Betriebsbereitschaft

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

Tabelle 2. Prüfliste für die Betriebsbereitschaft

Prüflistenpunkt	Weiterführende Informationen

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

Aktualisieren Sie die Datei `orb.properties`, um das Transportverhalten des Grids zu ändern. Sie finden die Datei `orb.properties` im Verzeichnis `java/jre/lib`.	„ORB-Eigenschaftendatei” auf Seite 207

Tabelle 2. Prüfliste für die Betriebsbereitschaft (Forts.)

Prüflistenpunkt	Weiterführende Informationen
Verwenden Sie Parameter im Script "startOgServer", insbesondere die folgenden: • Legen Sie die Einstellungen für den Heap-Speicher mit dem Parameter **-jvmArgs** fest. • Legen Sie den Anwendungsklassenpfad und Anwendungseigenschaften mit dem Parameter **-jvmArgs** fest. • Setzen Sie Parameter **-jvmArgs** für die Konfiguration der Agentenüberwachung. **Porteinstellungen** WebSphere eXtreme Scale muss für einige Transporte Kommunikationsports öffnen. Diese Ports werden alle dynamisch definiert. Wenn jedoch eine Firewall zwischen den Containern verwendet wird, müssen Sie die Ports angeben. Verwenden Sie die folgenden Portinformationen: **Listener-Port** Sie können das Argument **-listenerPort** verwenden, um den Port anzugeben, der für die Kommunikation zwischen Prozessen verwendet wird. **Stammgruppenport** Sie können das Argument **-haManagerPort** verwenden, um den Port anzugeben, der für die Fehlererkennung verwendet wird. Beachten Sie, dass Stammgruppen nicht zonenübergreifend kommunizieren müssen. Deshalb müssen Sie diesen Port unter Umständen nicht setzen, wenn die Firewall für alle Member einer einzigen Zone offen ist. **JMX-Serviceport** Sie können das Argument **-JMXServicePort** verwenden, um den Port anzugeben, den der JMX-Service verwenden soll. **SSL-Port** Wenn Sie `-Dcom.ibm.CSI.SSLPort=1234` als Argument mit **-jvmArgs** angeben, wird der SSL-Port auf 1234 gesetzt. Der SSL-Port ist der sichere Port-Peer zum Listener-Port. **Clientport** Wird nur im Katalogservice verwendet. Sie können diesen Wert mit dem Argument **-catalogServiceEndPoints** angeben. Der Wert für diesen Parameter muss im folgenden Format angegeben werden: `Servername:Hostname:Clientport:Peerport`	„Script "startOgServer"" auf Seite 235
Stellen Sie sicher, dass die Sicherheitseinstellungen ordnungsgemäß konfiguriert sind: • Transport (SSL) • Anwendung (Authentifizierung und Berechtigung) Zum Überprüfen Ihrer Sicherheitseinstellungen können Sie versuchen, einen zerstörerischen Client zu verwenden, der eine Verbindung zu Ihrer Konfiguration herstellt. Wenn beispielsweise die Einstellung "SSL-Required" konfiguriert ist, sollte ein Client, der die Einstellung TCP_IP hat, oder ein Client mit dem falschen Truststore nicht in der Lage sein, eine Verbindung zum Server herzustellen. Wenn eine Authentifizierung erforderlich ist, sollte ein Client ohne Berechtigungsnachweis (z. B. ohne Benutzer-ID und Kennwort) nicht in der Lage sein, eine Verbindung zum Server herzustellen. Wenn die Berechtigung zwingend erforderlich ist, sollte einem Client ohne Zugriffsberechtigung der Zugriff auf die Serverressourcen nicht erteilt werden.	Kapitel 9, „Implementierungsumgebung sichern", auf Seite 309
Wählen Sie aus, wie die Umgebung überwacht werden soll. • xsAdmin – Die JMX-Ports der Katalogserver müssen für das Tool "XSAdmin" sichtbar sein. Auch die Containerport müssen für einige Befehle, die Informationen von den Containern erfassen, zugänglich sein. • Sie können zwischen den folgenden Überwachungstools wählen: – Tivoli Enterprise Monitoring Agent – CA Wily Introscope – Hyperic HQ	• „Musterdienstprogramm "xsAdmin" verwenden" auf Seite 287 • „JMX-Sicherheit (Java Management Extensions)" auf Seite 321 • „Überwachung mit IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale" auf Seite 291 • „eXtreme Scale mit Hyperic HQ überwachen" auf Seite 301 • „eXtreme-Scale-Anwendungen mit CA Wily Introscope überwachen" auf Seite 297

Kapitel 3. Kapazitätsplanung

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

Grids, Partitionen und Shards

Ein verteiltes eXtreme-Scale-Grid ist in Partitionen unterteilt. Eine Partition enthält einen exklusiven Teil der Daten. Eine Partition setzt sich aus einem oder mehreren so genannten Shards (engl. Shard = Scherbe) zusammen: einem primären Shard und Replikat-Shards. Sie müssen keine Replikat-Shards in einer Partition verwenden, aber Replikat-Shards bieten eine hohe Verfügbarkeit. Unabhängig davon, ob Ihre Implementierung ein unabhängiges speicherinternes Daten-Grid oder ein speicherinterner Datenbankverarbeitungsbereich ist, ist der Datenzugriff in eXtreme Scale im Wesentlichen von den Shard-Konzepten abhängig.

Die Daten für eine Partition werden zur Laufzeit in einer Gruppe von Shards gespeichert. Diese Gruppe von Shards setzt sich aus einem primären Shard und unter Umständen einem oder mehreren Replikat-Shards zusammen. Ein Shard ist die kleinste Einheit, die eXtreme Scale in einer Java Virtual Machine hinzufügen oder entfernen kann.

Es sind zwei Verteilungsstrategien vorhanden: FIXED_PARTITIONS (Standardeinstellung) und PER_CONTAINER. Die folgende Beschreibung konzentriert sich auf die Verwendung der Strategie FIXED_PARTITIONS.

Anzahl der Shards

Wenn Ihre Umgebung beispielsweise zehn Partitionen mit einer Million Objekten ohne Replikate enthält, sind zehn Shards vorhanden, in denen jeweils 100.000 Objekte gespeichert sind. Wenn Sie diesem Szenario ein Replikat hinzufügen, ist in jeder Partition ein zusätzliches Shard vorhanden. In diesem Fall sind dann 20 Shards vorhanden, zehn primäre Shards und zehn Replikat-Shards. Und wieder sind in jedem dieser Shards 100.000 Objekte gespeichert. Jede Partition setzt sich aus einem primären Shard und einem oder mehreren (N) Replikat-Shards zusammen. Die Festlegung der optimalen Shard-Anzahl ist ein kritischer Faktor. Wenn Sie eine geringe Anzahl an Shards konfigurieren, werden die Daten nicht gleichmäßig auf die Shards verteilt, was zu Fehlern wegen Speicherengpässen und Problemen durch Prozessüberlastung führt. Sie müssen bei der Skalierung mindestens zehn Shards für jede JVM festlegen. Wenn Sie das Grid implementieren, verwenden Sie möglicherweise eine höhere Anzahl an Partitionen.

Anzahl der Shards pro JVM

Szenario mit einer geringen Anzahl an Shards für jede JVM

Daten werden in einer JVM in Shard-Einheiten hinzugefügt und entfernt. Shards werden nie unterteilt. Wenn 10 GB Daten und 20 Shards zum Speichern dieser

Daten vorhanden sind, enthält jedes Shard durchschnittlich 500 MB Daten. Wenn das Grid auf neun JVMs verteilt ist, hat jede JVM durchschnittlich zwei Shards. Da 20 nicht durch 9 teilbar ist, haben einige JVMs drei Shards. Die Verteilung ist wie folgt:

- 7 JVMs mit 2 Shards
- 2 JVMs mit 3 Shards

Da jedes Shard 500 MB Daten enthält, ist die Verteilung der Daten unsymmetrisch. Die sieben JMVs mit zwei Shards enthalten jeweils 1 GB Daten. Die beiden JVMs mit drei Shards enthalten jeweils 50 % mehr Daten (oder 1,5 GB), was eine sehr viel höhere Speicherlast darstellt. Da diese beiden JVMs drei Shards haben, empfangen sie auch 50 % mehr Anforderungen für ihre Daten. Deshalb führt eine geringe Anzahl an Shards für jede JVM zu einem Ungleichgewicht. Zur Verbesserung der Leistung erhöhen Sie die Anzahl der Shards für jede JVM.

Szenario mit einer höheren Anzahl an Shards pro JVM

In diesem Szenario wird eine sehr viel höhere Anzahl an Shards verwendet. Es gibt 101 Shards mit 9 JVMs für 10 GB Daten. In diesem Fall werden in jedem Shard 99 MB Daten gespeichert. Die Shards sind wie folgt auf die JVMs verteilt:

- 7 JVMs mit 11 Shards
- 2 JVMs mit 12 Shards

Die beiden JVMs mit jeweils 12 Shards enthalten jetzt nur 99 MB mehr Daten als die anderen Shards. Dies ergibt eine Differenz von 9 %. In diesem Szenario ist die Last gleichmäßiger verteilt als in dem Szenario mit einer geringeren Anzahl an Shards, wo die Differenz bei 50 % liegt. Vom Standpunkt der Prozessorauslastung betrachtet, fällt auf die beiden JVMs mit den jeweils 12 Shards im Vergleich mit den sieben JVMs mit jeweils 11 Shards nur 9 % mehr Arbeit. Durch die Erhöhung der Shard-Anzahl in jeder JVM wird die Daten- und Prozessorauslastung fair und gleichmäßig verteilt.

Verwenden Sie 10 Shards für jede JVM, wenn Sie Ihr System in maximaler Größe bzw. die Ausführung der maximalen Anzahl an JVMs in Ihrem System planen.

Speicher dimensionieren und Partitionsanzahl berechnen

Sie können die für Ihre Konfiguration benötige Speicherkapazität und die benötige Partitionsanzahl berechnen.

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

Zur Dimensionierung des benötigten Speichers laden Sie Ihre Anwendungsdaten in eine einzige JVM. Wenn die Heap-Speicherbelegung einen Wert von 60 % erreicht, notieren Sie die Anzahl der verwendeten Objekte. Diese Zahl ist die maximal empfohlene Objektanzahl für jede Ihrer JVMs. Für eine möglichst genaue Dimensionierung sollten Sie realistische Daten verwenden und alle definierten Indizes einbeziehen, weil Indizes auch Speicher belegen. Die beste Methode für die Speicherdimensionierung ist die Ausführung einer ausführlichen Garbage-Collection (mit verbosegc), da diese Ausgabe Ihnen die Zahlen nach der Garbage-Collec-

tion liefert. Sie können die Heap-Speicherbelegung jederzeit über MBeans oder über das Programm abfragen, aber diese Abfragen liefern Ihnen nur eine aktuelle Momentaufnahme des Heap-Speichers, die nicht erfassten Garbage (fehlerhafte Daten) enthalten kann. Deshalb ist die Verwendung dieser Methode keine genaue Indikation auf den belegten Speicher.

Konfiguration vertikal skalieren

Anzahl der Shards pro Partition (numShardsPerPartition)

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

```
numShardsPerPartition = 1 + Gesamtanzahl_der_Replikate
```

Anzahl der JVMs (minNumJVMs)

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

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

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

Anzahl der Shards (numShards)

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

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

Anzahl der Partitionen

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

```
numPartitions = numShards / numShardsPerPartition
```

Skalierungsbeispiel

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

- 500 JVMs zu Speichern der endgültigen Anzahl an Einträgen
- 5000 Shards (500 Java Virtual Machines mal 10)

- 2500 Partitionen bzw. 2503, da dies die nächst höhere Primzahl ist (5000 Shards, geteilt durch zwei für primäre und Replikat-Shards)

Anfangskonfiguration

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

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

Erreichen der maximalen Anzahl an Java Virtual Machines

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

Anzahl der Server

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

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

CPU-Dimensionierung pro Partition für Transaktionen

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

Die Prozessorkosten setzen sich wie folgt zusammen:

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

Java Virtual Machines pro Server

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

Versuchen Sie die Prozessorauslastung zu sättigen, um das Ausgangsniveaus zu berechnen. Sollte dies nicht möglich sein, ist das Netz wahrscheinlich gesättigt. Wenn das Netz gesättigt ist, fügen Sie weitere Netzkarten hinzu, und bringen Sie die Java Virtual Machines auf den verschiedenen Netzkarten in Umlauf.

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

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

Dimensionierung der CPUs für parallele Transaktionen

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

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

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

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

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

Zusammenfassung

Für diese Dimensionierung stehen Ihnen drei Messwerte zur Verfügung:

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

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

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

Kapitel 4. WebSphere eXtreme Scale installieren und implementieren

WebSphere eXtreme Scale ist ein speicherinternes Daten-Grid, das Sie verwenden können, um Anwendungsdaten und Geschäftslogik in mehreren Servern zu partitionieren, zu replizieren und zu verwalten.

Vorbereitungen

- Stellen Sie fest, wie WebSphere eXtreme Scale in Ihre aktuelle Topologie passt. Weitere Informationen hierzu finden Sie in der Übersicht über die Architektur und Topologie von WebSphere eXtreme Scale im Handbuch *Produktübersicht*.
- Vergewissern Sie sich, dass Ihre Umgebung die Voraussetzungen für die Installation von eXtreme Scale erfüllt. Weitere Informationen finden Sie im Abschnitt „Hardware- und Softwarevoraussetzungen" auf Seite 7.

Warum und wann dieser Vorgang ausgeführt wird

Unterstützte Umgebungen

Sie müssen eXtreme Scale nicht unter einer bestimmten Version des Betriebssystems installieren und implementieren. Jede Java-SE- und jede Java-EE-Installation setzt andere Betriebssystemversionen oder -Fixes voraus.

Sie können das Produkt in Java-EE- und Java-SE-Umgebungen installieren und implementieren. Sie können die Clientkomponente auch direkt mit Java-EE-Anwendungen ohne Integration in WebSphere Application Server bündeln. eXtreme Scale unterstützt Java Runtime Environment (JRE) Version 1.4.2 und höher und WebSphere Application Server Version 6.0.2 und höher.

Eigenständiges eXtreme Scale

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

Integration des Produkts in WebSphere Application Server oder WebSphere Application Server Network Deployment

Sie können eXtreme Scale in eine vorhandene Installation von WebSphere Application Server oder WebSphere Application Server Network Deployment installieren und integrieren. Sie können den eXtreme-Scale-Client und den eXtreme-Scale-Server installieren, oder Sie können auch nur den Client installieren.

Profile erstellen und erweitern

Erstellen und erweitern Sie Profile für die Verwendung der Features von eXtreme Scale.Wenn Sie WebSphere Application Server Version 6.1 oder Version 7.0 ausführen, können Sie das PMT-Plug-in (Profile Management Tool) oder den Befehl

manageprofiles verwenden. Wenn Sie WebSphere Application Server Version 6.0.2 ausführen, müssen Sie zum Erstellen und Erweitern von Profilen den Befehl wasprofile verwenden.

Wartungspakete anwenden

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

Migration auf WebSphere eXtreme Scale Version 7.0

Wenn Sie ein Upgrade von Version 6.1.0.x auf Version 7.0 durchführen möchten, müssen Sie alle geänderten Produktscriptdateien mit den neuen Produktscriptdateien zusammenführen, um Ihre Änderungen beizubehalten.

Vorbereitungen

Vergewissern Sie sich, dass Ihre Systeme die Mindestvoraussetzungen für die Produktversionen erfüllen, die Sie migrieren und installieren möchten. Weitere Informationen finden Sie im Abschnitt „Hardware- und Softwarevoraussetzungen" auf Seite 7.

Warum und wann dieser Vorgang ausgeführt wird

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

Tipp: Wenn Sie die Scriptdateien, die mit dem Produkt installiert werden, nicht ändern, müssen Sie die folgenden Migrationsschritte nicht ausführen. Stattdessen können Sie das Upgrade auf Version 7.0 durchführen, indem Sie die vorherige Version deinstallieren und die neue Version in demselben Verzeichnis installieren.

1. Stoppen Sie alle Prozesse, die eXtreme Scale verwenden.
   - Informationen zum Stoppen aller Prozess, die in Ihrer eigenständigen eXtreme-Scale-Umgebung aktiv sind, finden Sie im Abschnitt „Eigenständige eXtreme-Scale-Server stoppen" auf Seite 239.
   - Informationen zum Stoppen aller Prozesse, die in Ihrer Umgebung von WebSphere Application Server oder WebSphere Application Server Network Deployment aktiv sind, finden Sie im Information-Center-Artikel Befehlszeilendienstprogramme.
2. Speichern Sie alle geänderten Scripts aus Ihrem aktuellen Installationsverzeichnis in einem temporären Verzeichnis.
3. Deinstallieren Sie das Produkt.
4. Installieren Sie eXtreme Scale Version 7.0. Weitere Informationen finden Sie im Abschnitt Kapitel 4, „WebSphere eXtreme Scale installieren und implementieren", auf Seite 27.
5. Fügen Sie Ihre Änderungen aus den Dateien im temporären Verzeichnis in die neuen Produktscriptdateien im Verzeichnis `/bin` ein.
6. Starten Sie alle Prozesse von eXtreme Scale, um mit der Verwendung des Produkts zu beginnen. Weitere Informationen finden Sie im Abschnitt Kapitel 7, „Umgebung verwalten", auf Seite 225.

Eigenständige Version von WebSphere eXtreme Scale installieren

Sie können die eigenständige Version von WebSphere eXtreme Scale in einer Umgebung installieren, die weder WebSphere Application Server noch WebSphere Application Server Network Deployment enthält.

Vorbereitungen

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

  Anmerkung: Wenn eine vorherige Version von eXtreme Scale oder die Komponente "ObjectGrid" in dem Verzeichnis vorhanden ist, das Sie für die Installation der Version 7.0 angeben, wird das Produkt nicht installiert. Sie können ein anderes Installationsverzeichnis auswählen oder die Installation abbrechen. Deinstallieren Sie anschließend die vorherige Installation, und führen Sie den Assistenten erneut aus.
- Für eine bessere Leistung und Servicefreundlichkeit laden Sie ein IBM Developer Kit von developerWorks herunter und installieren es.

  Einschränkung: Wenn Sie Hardware eines unabhängigen Anbieters verwenden, wählen Sie eine der folgenden Optionen aus, um ein Developer-Kit herunterzuladen und zu installieren:
  - Sun JDK herunterladen
  - JDK oder JRE eines unabhängigen Softwareanbieters herunterladen

Warum und wann dieser Vorgang ausgeführt wird

Wenn Sie das Produkt eigenständig installieren, installieren Sie den eXtreme-Scale-Client und den eXtreme-Scale-Server unabhängig voneinander. Server- und Clientprozesse greifen deshalb auf alle erforderlichen Ressourcen lokal zu. Sie können eXtreme Scale über Scripts und JAR-Dateien auch in vorhandene Java Platform, Standard Edition-Anwendungen integrieren.

In der folgenden Tabelle sind die JAR-Dateien aufgelistet, die in der Installation enthalten sind.

Tabelle 3. Laufzeitdateien im Installationsverzeichnis /ObjectGrid/lib

Dateiname	Umgebung	Beschreibung
cglib.jar	Lokal, Client und Server	Die Datei cglib.jar wird von der Dienstprogrammfunktion "cglib" gelesen, wenn Sie den Kopiermodus "copy-on-write" (Erstellen einer Kopie beim Schreiben) oder EntityManager verwenden, um die Änderungen einer Entität zu verfolgen. Die Datei wird automatisch in die Serverlaufzeitumgebung eingeschlossen, wenn Sie die bereitgestellten Scripts verwenden. Fügen Sie diese Datei der Client- oder lokalen Laufzeitumgebung hinzu.
objectgrid.jar	Lokal, Client und Server	Die Datei objectgrid.jar wird von der Serverlaufzeitumgebung von Java Platform, Standard Edition Version 1.4.2 und höher verwendet. Die Datei wird automatisch in die Serverlaufzeitumgebung eingeschlossen, wenn Sie die bereitgestellten Scripts verwenden.
ogagent.jar	Lokal, Client und Server	Die Datei ogagent.jar enthält die Laufzeitklassen, die für die Ausführung des Java-Instrumentierungsagenten erforderlich sind, der mit der API "EntityManager" verwendet wird.
ogclient.jar	Lokal und Client	Die Datei ogclient.jar enthält nur die lokale Laufzeitumgebung und die Clientlaufzeitumgebung. Sie können diese Datei mit Java SE Version 1.4.2 und höher verwenden.
wsogclient.jar	Lokal und Client	Die Datei wsogclient.jar wird installiert, wenn Sie das Produkt in einer Umgebung mit WebSphere Application Server Version 6.0.2 oder höher installieren. Die Datei enthält nur die lokale und die Clientlaufzeitumgebung.
wxsdynacache.jar	Nur Server	Die Datei wxsdynacache.jar enthält die erforderlichen Klassen für den dynamischen Cacheprovider. Die Datei wird automatisch in die Serverlaufzeitumgebung eingeschlossen, wenn Sie die bereitgestellten Scripts verwenden. **Achtung:** Sie finden die Datei im Verzeichnis ObjectGrid/dynacache/lib.

1. Verwenden Sie den Assistenten, um die Installation durchzuführen. Führen Sie das folgende Script aus, um den Assistenten zu starten:
   - *DVD-Stammverzeichnis*/install
   - *DVD-Stammverzeichnis*\install.bat
2. Folgen Sie den Eingabeaufforderungen im Assistenten, und klicken Sie auf **Beenden**, um die Installation abzuschließen.

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

Nächste Maßnahme

Konfigurieren Sie Ihre Clientanwendungsprozesse und Serverprozesse. Nähere Informationen hierzu finden Sie im Abschnitt Kapitel 6, „WebSphere eXtreme Scale konfigurieren", auf Seite 63.

Zugehörige Verweise

„Installationsparameter" auf Seite 53
Geben Sie Parameter in der Befehlszeile an, um Ihre Produktinstallation anzupassen und zu konfigurieren.

Object Request Broker mit WebSphere eXtreme Scale-Prozessen verwenden

Sie können WebSphere eXtreme Scale mit Anwendungen verwenden, die den Object Request Broker (ORB) direkt in Umgebungen ohne WebSphere Application Server oder WebSphere Application Server Network Deployment verwenden.

Vorbereitungen

Wenn Sie den ORB in demselben Prozess wie eXtreme Scale verwenden, müssen Sie bei der Ausführung von Anwendungen oder anderen Komponenten und Frameworks, die nicht mit eXtreme Scale bereitgestellt werden, möglicherweise zusätzliche Tasks ausführen, um sicherzustellen, dass eXtreme Scale ordnungsgemäß in Ihrer Umgebung ausgeführt wird.

Warum und wann dieser Vorgang ausgeführt wird

Fügen Sie der Datei `orb.properties` die Eigenschaft "ObjectGridInitializer" hinzu, um die Verwendung des ORB in Ihrer Umgebung zu initialisieren. Verwenden Sie den ORB, um die Kommunikation zwischen eXtreme-Scale-Prozessen und anderen Prozessen in Ihrer Umgebung zu aktivieren. Sie finden die Datei `orb.properties` im Verzeichnis `java/jre/lib`. Beschreibungen der Eigenschaften und Einstellungen finden Sie im Abschnitt „ORB-Eigenschaftendatei" auf Seite 207.

Geben Sie die folgende Zeile ein, und speichern Sie Ihre Änderungen:

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

Ergebnisse

eXtreme Scale initialisiert den ORB ordnungsgemäß und koexistiert mit anderen Anwendungen, für die der ORB aktiviert ist.

Wenn Sie eine angepasste Version des ORB mit eXtreme Scale verwenden möchten, lesen Sie den Abschnitt „Angepassten Object Request Broker konfigurieren" auf Seite 56.

Zugehörige Verweise

„ORB-Eigenschaftendatei" auf Seite 207
Die Datei `orb.properties` wird für die Übergabe der vom Object Request Broker (ORB) verwendeten Eigenschaften verwendet, um das Transportverhalten des Grids zu ändern.

Angepasste Eigenschaften des Object Request Broker (ORB)

WebSphere eXtreme Scale in WebSphere Application Server integrieren

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

Vorbereitungen

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

Warum und wann dieser Vorgang ausgeführt wird

Integrieren Sie eXtreme Scale in WebSphere Application Server bzw. WebSphere Application Server Network Deployment, um die Features von eXtreme Scale auf Ihre Java-EE-Anwendungen anzuwenden. Java-EE-Anwendungen enthalten eXtreme-Scale-Grids und greifen auf die Grids über eine Clientverbindung zu.

In der folgenden Tabelle sind die JAR-Dateien aufgelistet, die in der Installation enthalten sind.

Tabelle 4. Laufzeitdateien im Installationsverzeichnis /`lib`

Dateiname	Umgebung	Beschreibung
`cglib.jar`	Lokal, Client und Server	Die Datei `cglib.jar` wird von der Dienstprogrammfunktion "cglib" gelesen, wenn Sie den Kopiermodus "copy-on-write" (Erstellen einer Kopie beim Schreiben) oder die API "EntityManager" verwenden, um die Änderungen einer Entität zu verfolgen.
`ogagent.jar`	Lokal, Client und Server	Die Datei `ogagent.jar` enthält die Laufzeitklassen, die für die Ausführung des Java-Instrumentierungsagenten erforderlich sind, der mit der API "EntityManager" verwendet wird.

Tabelle 4. Laufzeitdateien im Installationsverzeichnis /lib (Forts.)

Dateiname	Umgebung	Beschreibung
wsobjectgrid.jar	Lokal, Client und Server	Die Datei wsobjectgrid.jar enthält die lokale, die Client- und die Serverlaufzeitumgebung von eXtreme Scale.
wsogclient.jar	Lokal und Client	Die Datei wsogclient.jar wird installiert, wenn Sie das Produkt in einer Umgebung mit WebSphere Application Server Version 6.0.2 oder höher installieren. Die Datei enthält nur die lokale und die Clientlaufzeitumgebung.
wxsdynacache.jar	Nur Server	Die Datei wxsdynacache.jar enthält die erforderlichen Klassen für den dynamischen Cacheprovider.

1. Verwenden Sie den Assistenten, um die Installation durchzuführen. Führen Sie das folgende Script aus, um den Assistenten zu starten:
   - *DVD-Stammverzeichnis*/install
   - *DVD-Stammverzeichnis*\install.bat
2. Folgen Sie den Eingabeaufforderungen im Assistenten.

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

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

Nächste Maßnahme

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

Implementieren Sie Ihre Anwendung, starten Sie einen Katalogservice, und starten Sie die Container in Ihrer Umgebung mit WebSphere Application Server. Weitere Informationen hierzu finden Sie im Abschnitt „WebSphere eXtreme Scale mit WebSphere Application Server verwalten" auf Seite 242.

Zugehörige Verweise

„Installationsparameter" auf Seite 53
Geben Sie Parameter in der Befehlszeile an, um Ihre Produktinstallation anzupassen und zu konfigurieren.

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

Verwenden Sie das IBM Installation-Factory-Plug-in für WebSphere eXtreme Scale, um ein angepasstes Installationspaket (CIP, Customized Installation Package) oder ein integriertes Installationspaket (IIP, Integrated Installation Package) zu erstellen. Ein angepasstes Installationspaket enthält ein Installationspaket für ein Einzelprodukt und verschiedene optionale Assets. Ein integriertes Installationspaket kombiniert ein oder mehrere Installationspakete zu einem einzigen Installations-Workflow, den Sie entwerfen können.

Vorbereitungen

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

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

Warum und wann dieser Vorgang ausgeführt wird

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

Build-Definitionsdatei

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

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

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

Wenn Sie nur die Build-Definitionsdatei erstellen möchten, können Sie das Tool mit der Befehlszeilenschnittstelle ausführen, um das angepasste Paket außerhalb der grafischen Benutzerschnittstelle zu generieren. Weitere Informationen finden Sie im Abschnitt „Angepasstes oder integriertes Installationspaket im unbeaufsichtigten Modus installieren” auf Seite 40.

Build-Definitionsdatei erstellen und ein angepasstes Installationspaket erstellen

Das IBM Installation Factory-Plug-in für WebSphere eXtreme Scale generiert ein angepasstes Installationspaket entsprechend den Details, die Sie in der Build-

Definitionsdatei angeben. Die Build-Definition gibt das zu installierende Produktpaket, die Position des angepassten Installationspakets, die in die Installation einzuschließenden Wartungspakete, die Installationsscriptdateien und alle zusätzlichen Dateien an, die in das angepasste Installationspaket eingeschlossen werden sollen.

Warum und wann dieser Vorgang ausgeführt wird

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

1. Führen Sie das folgende Script im Verzeichnis *IF_HOME*/bin aus, um Installation Factory zu starten:
   - ifgui.sh
   - ifgui.bat

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

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

Ergebnisse

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

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

Nächste Maßnahme

Installieren Sie das angepasste Installationspaket.

Ein angepasstes Installationspaket installieren:

Sie können den Produktinstallationsprozess vereinfachen, indem Sie ein angepasstes Installationspaket (CIP, Customized Installation Package) installieren. Ein angepasstes Installationspaket ist ein Installations-Image für ein Einzelprodukt, das mindestens ein Wartungspaket, Konfigurationsscripts und andere Dateien enthalten kann.

Vorbereitungen

Damit Sie ein integriertes Installationspaket installieren können, müssen Sie zuerst eine Build-Definitionsdatei erstellen, in der Sie die in das integrierte Installationspaket einzuschließenden Optionen angeben. Weitere Informationen finden Sie im

Abschnitt „Build-Definitionsdatei erstellen und ein angepasstes Installationspaket erstellen" auf Seite 33.

Warum und wann dieser Vorgang ausgeführt wird

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

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

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

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

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

Ergebnisse

Sie haben das angepasste Installationspaket ordnungsgemäß installiert.

Nächste Maßnahme

Wenn Sie WebSphere Application Server Version 6.1 oder Version 7.0 ausführen, können Sie das PMT-Plug-in (Profile Management Tool) oder den Befehl manageprofiles verwenden, um Profile zu erstellen und zu erweitern. Wenn Sie WebSphere Application Server Version 6.0.2 ausführen, müssen Sie zum Erstellen und Erweitern von Profilen den Befehl wasprofile verwenden. Weitere Informationen finden Sie im Abschnitt „Profile für WebSphere eXtreme Scale erstellen und erweitern" auf Seite 42.

Wenn Sie Profile für eXtreme Scale während des Installationsprozesses erweitert haben, können Sie Anwendungen implementieren, einen Katalogservice starten und die Container in Ihrer Umgebung von WebSphere Application Server starten. Weitere Informationen hierzu finden Sie im Abschnitt „WebSphere eXtreme Scale mit WebSphere Application Server verwalten" auf Seite 242.

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

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

Vorbereitungen

Erstellen Sie eine Build-Definitionsdatei, um die in das angepasste Installationspaket einzuschließenden Optionen anzugeben. Weitere Informationen finden Sie im Abschnitt „Build-Definitionsdatei erstellen und ein angepasstes Installationspaket erstellen" auf Seite 33.

Warum und wann dieser Vorgang ausgeführt wird

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

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

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

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

Ergebnisse

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

CIP-Aktualisierungen aus einer vorhandenen Produktinstallation deinstallieren:

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

Vorbereitungen

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

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

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

   Angepasstes Installationspaket

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

Ergebnisse

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

Build-Definitionsdatei erstellen und angepasstes Integrationspaket generieren

Das IBM Installation-Factory-Plug-in für WebSphere eXtreme Scale generiert ein integriertes Installationspaket (IIP, Integrated Installation Package), basierend auf den in der Build-Definitionsdatei enthaltenen Eigenschaften, z. B. den in das integrierte Installationspaket einzuschließenden Installationspaketen, der Reihenfolge, in der Installation Factory die Pakete installiert, und der Position des integrierten Installationspakets.

Warum und wann dieser Vorgang ausgeführt wird

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

1. Führen Sie das folgende Script im Verzeichnis *IF_HOME*`/bin` aus, um Installation Factory zu starten:
   - `ifgui.sh`

- `ifgui.bat`

2. Klicken Sie auf das Symbol **Neues integriertes Installationspaket erstellen**, um den Assistenten für Build-Definition zu starten.
3. Folgen Sie den Eingabeaufforderungen im Assistenten.
   a. Wählen Sie in der Anzeige "Integriertes Installationspaket erstellen" in der Liste ein unterstütztes Installationspaket aus, und klicken Sie auf **Installationsprogramm hinzufügen**, um das Installationspaket dem integrierten Installationspaket hinzuzufügen. Es erscheint eine Anzeige, in der der Paketname, die Paketkennung und die Paketeigenschaften angezeigt werden. Wenn Sie bestimmte Informationen zum ausgewählten Paket anzeigen möchten, klicken Sie auf **Informationen zum Installationspaket anzeigen**. Klicken Sie auf **Ändern**, um den Verzeichnispfad zum Installationspaket für jedes Betriebssystem einzugeben. Wenn Sie ein Installationspaket für WebSphere Extended Deployment hinzufügen, wählen Sie das Kontrollkästchen aus, das Ihnen die Möglichkeit gibt, dasselbe Paket für alle unterstützten Betriebssysteme zu verwenden. Klicken Sie auf **OK**, und kehren Sie in die Anzeige "Integriertes Installationspaket erstellen" zurück. Es wird standardmäßig ein Aufruf erstellt.
      - Wenn Sie den Verzeichnispfad zu einem Installationspaket ändern möchten, wählen Sie das Paket in der Liste "In diesem integrierten Installationspaket verwendete Installationspakete" aus, und klicken Sie auf **Ändern**.
      - Wenn Sie einen Aufruf ändern möchten, wählen Sie den Aufruf aus, und klicken Sie auf **Ändern**. Geben Sie die Standardinstallationsposition für den Aufruf auf jedem Betriebssystem an. Geben Sie die Position zur Antwortdatei an, wenn Sie eine unbeaufsichtigte Installation als Standardinstallationsmodus auswählen.
      - Klicken Sie auf **Aufruf hinzufügen**, um dem Installationspaket einen Aufrufbeitrag hinzuzufügen. Es erscheint eine Anzeige, in der Sie die Eigenschaften für den Aufruf eingeben können.
      - Klicken Sie auf **Entfernen**, um Installationspakete oder Aufrufe zu entfernen.
4. Sehen Sie sich die Zusammenfassung der von Ihnen ausgewählten Optionen an, wählen Sie die Option **Build-Definitionsdatei speichern und integriertes Installationspaket generieren** aus, und klicken Sie anschließend auf **Fertig stellen**.

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

Ergebnisse

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

Nächste Maßnahme

Installieren Sie das integrierte Installationspaket.

Integriertes Installationspaket installieren:

Verwenden Sie das IBM Installation Factory-Plug-in für WebSphere eXtreme Scale, um ein integriertes Installationspaket (IIP, Integrated Installation Package) zu

installieren. Ein integriertes Installationspaket kombiniert ein oder mehrere Installationspakete zu einem einzigen Workflow, den Sie entwerfen können.

Vorbereitungen

Damit Sie ein integriertes Installationspaket installieren können, müssen Sie zuerst eine Build-Definitionsdatei erstellen, in der Sie die in das integrierte Installationspaket einzuschließenden Optionen angeben. Weitere Informationen finden Sie im Abschnitt „Build-Definitionsdatei erstellen und angepasstes Integrationspaket generieren" auf Seite 37.

Warum und wann dieser Vorgang ausgeführt wird

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

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

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

   Optional: Auf der Assistentenseite werden zwei Schaltflächen **Protokoll anzeigen** angezeigt. Wenn Sie die Protokolle der einzelnen Pakete anzeigen möchten, klicken Sie auf die Schaltfläche **Protokoll anzeigen**, die neben der Tabelle angezeigt wird, in der die Installationspakete aufgelistet sind. Klicken Sie auf die Schaltfläche **Protokoll anzeigen**, die neben den Statusinformationen angezeigt wird, um die allgemeinen Protokolldetails des integrierten Installationspakets anzuzeigen.
3. Wählen Sie die auszuführenden Installationspakete aus, und klicken Sie auf **Installieren**. Es wird eine nach Aufruf sortierte Liste aller Beiträge angezeigt, die im integrierten Installationspaket enthalten sind. Wenn ein Beitragsaufruf während der Installation nicht ausgeführt werden soll, wählen Sie das Kontrollkästchen neben dem Feld **Installationsname** ab.

Ergebnisse

Sie haben ein integriertes Installationspaket erfolgreich installiert.

Eine vorhandene Build-Definitionsdatei für ein integriertes Installationspaket ändern:

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

Warum und wann dieser Vorgang ausgeführt wird

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

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

     Wichtig: Wenn Sie ein unterstütztes Betriebssystem hinzugefügt und die Eigenschaften des Installationspaket im integrierten Installationspaket nicht aktualisiert haben, empfangen Sie eine Warnung, in der Sie darauf hingewiesen werden, dass die ausgewählten Beiträge keine Installationspakete enthalten, die für alle Betriebssysteme angegeben sind, die vom integrierten Installationspaket unterstützt werden. Klicken Sie auf **Ja**, um fortzufahren, oder klicken Sie auf **Nein**, um das Installationspaket zu bearbeiten.
4. Sehen Sie sich die Zusammenfassung der von Ihnen ausgewählten Optionen an, wählen Sie die Option **Build-Definitionsdatei speichern und integriertes Installationspaket generieren** aus, und klicken Sie anschließend auf **Fertig stellen**.

Angepasstes oder integriertes Installationspaket im unbeaufsichtigten Modus installieren

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

Vorbereitungen

Erstellen Sie die Build-Definitionsdatei für das angepasste Installationspaket bzw. das integrierte Installationspaket. Weitere Informationen finden Sie im Abschnitt „Build-Definitionsdatei erstellen und ein angepasstes Installationspaket erstellen" auf Seite 33.

Warum und wann dieser Vorgang ausgeführt wird

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

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

1. Optional: Wenn Sie sich für die Installation des angepassten Installationspakets bzw. integrierten Installationspakets über eine Antwortdatei entscheiden, passen Sie zuerst die Datei an.
   a. Kopieren Sie die Antwortdatei `wxssetup.response.txt` von der Produkt-DVD auf Ihr Plattenlaufwerk.
   b. Öffnen Sie die Antwortdatei mit einem Texteditor Ihrer Wahl, und bearbeiten Sie sie. Die Datei enthält Kommentare, die Sie beim Konfigurationsprozess unterstützen. Sie muss die folgenden Parameter enthalten:
      - die Lizenzvereinbarung,
      - die Position der Produktinstallation,

      Tipp: Das Installationsprogramm verwendet die Position, die Sie für Ihre Installation auswählen, um zu bestimmen, wo Ihre Instanz von WebSphere Application Server installiert ist. Wenn Sie die Installation auf einem Knoten mit mehreren Instanzen von WebSphere Application Server durchführen, müssen Sie die Position klar definieren.
   c. Führen Sie das folgende Script aus, um die angepasste Antwortdatei zu starten.
      - `install -options /absoluter_Pfad/Antwortdatei.txt -silent`
      - `install.bat -options C:\Laufwerkspfad\Antwortdatei.txt -silent`
2. Optional: Wenn Sie sich für die Installation des angepassten bzw. integrierten Installationspakets durch Übergabe bestimmter Parameter in der Befehlszeile entscheiden, führen Sie das folgende Script aus, um die Installation zu starten:
   - `install -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=Installationsposition`
   - `install.bat -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=Installationsposition`

Installationsposition steht für die Position der vorhandenen Installation von WebSphere Application Server.

3. Suchen Sie in den während der Installation erstellten Protokolle nach Fehlern oder Installationsfehlern.

Ergebnisse

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

Nächste Maßnahme

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

Wenn Sie Profile für eXtreme Scale während des Installationsprozesses erweitert haben, können Sie Anwendungen implementieren, einen Katalogservice starten und die Container in Ihrer Umgebung von WebSphere Application Server starten. Weitere Informationen hierzu finden Sie im Abschnitt „WebSphere eXtreme Scale mit WebSphere Application Server verwalten" auf Seite 242.

Profile für WebSphere eXtreme Scale erstellen und erweitern

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

Vorbereitungen

Installieren Sie WebSphere eXtreme Scale. Weitere Informationen finden Sie im Abschnitt „WebSphere eXtreme Scale in WebSphere Application Server integrieren" auf Seite 31.

Warum und wann dieser Vorgang ausgeführt wird

Ausführung in WebSphere Application Server Version 6.0.2

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

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

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

Ausführung in WebSphere Application Server Version 6.1 oder Version 7.0

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

Nächste Maßnahme

Abhängig davon, welche Aufgabe Sie ausführen möchten, starten Sie dafür die Konsole "Erste Schritte", um Unterstützung bei der Konfiguration und beim Test Ihrer Produktumgebung zu erhalten. Alternativ dazu können Sie jede vorher aufgeführte Aufgabe erneut ausführen, um weitere Profile zu erstellen oder zu erweitern.

Zugehörige Verweise

„Installationsparameter" auf Seite 53
Geben Sie Parameter in der Befehlszeile an, um Ihre Produktinstallation anzupassen und zu konfigurieren.

„Befehl "manageprofiles"" auf Seite 45
Sie können das Dienstprogramm manageprofiles verwenden, um Profile mit der eXtreme-Scale-Schablone zu erstellen und vorhandene Anwendungsserverprofile mit den eXtreme-Scale-Erweiterungsschablonen zu erweitern bzw. deren Erweiterung aufzuheben. Zur Verwendung der Produktfeatures muss Ihre Umgebung mindestens ein Profil enthalten, das für das Produkt erweitert wurde.

Grafische Benutzerschnittstelle für die Erstellung von Profilen verwenden

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

Vorbereitungen

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

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

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

Warum und wann dieser Vorgang ausgeführt wird

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

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

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

Die Seite "Aktionsauswahl" wird nur angezeigt, wenn mindestens ein Profil und die Erweiterungsschablonen vorhanden sind.

Nächste Maßnahme

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

Starten Sie einen Katalogservice, starten Sie Container, und konfigurieren Sie TCP-Ports in Ihrer Umgebung mit WebSphere Application Server. Weitere Informationen hierzu finden Sie im Abschnitt „WebSphere eXtreme Scale mit WebSphere Application Server verwalten" auf Seite 242.

Zugehörige Verweise

„Befehl "manageprofiles"" auf Seite 45
Sie können das Dienstprogramm manageprofiles verwenden, um Profile mit der eXtreme-Scale-Schablone zu erstellen und vorhandene Anwendungsserverprofile mit den eXtreme-Scale-Erweiterungsschablonen zu erweitern bzw. deren Erweiterung aufzuheben. Zur Verwendung der Produktfeatures muss Ihre Umgebung mindestens ein Profil enthalten, das für das Produkt erweitert wurde.

Grafische Benutzerschnittstelle für die Erweiterung von Profilen verwenden

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

Vorbereitungen

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

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

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

Warum und wann dieser Vorgang ausgeführt wird

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

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

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

- Wählen Sie in der Konsole "Erste Schritte" die Option **Profile Management Tool** aus.
- Rufen Sie Profile Management Tool über das Menü **Start** auf.

- Führen Sie das Script "./pmt.sh|bat" im Verzeichnis *Installationsstammverzeichnis*/bin/ProfileManagement aus.

Nächste Maßnahme

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

Starten Sie einen Katalogservice, starten Sie Container, und konfigurieren Sie TCP-Ports in Ihrer Umgebung mit WebSphere Application Server. Weitere Informationen hierzu finden Sie im Abschnitt „WebSphere eXtreme Scale mit WebSphere Application Server verwalten" auf Seite 242.

Zugehörige Verweise

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

Befehl "manageprofiles"

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

- Bevor Sie Profile erstellen und erweitern können, müssen Sie eXtreme Scale installieren. Weitere Informationen finden Sie im Abschnitt „WebSphere eXtreme Scale in WebSphere Application Server integrieren" auf Seite 31.
- Wenn Ihre Umgebung WebSphere Application Server Version 6.0.2 enthält, müssen Sie den Befehl wasprofile verwenden, um Profile für eXtreme Scale zu erstellen und zu erweitern, wie im folgenden Beispiel gezeigt wird:

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

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

Zweck

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

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

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

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

Position

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

Verwendung

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

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

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

Deployment-Manager-Profil erstellen

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

Parameter

-create
Erstellt ein Profil. (Erforderlich)

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

Verwenden Sie das folgende Format:

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

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":

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

- Mit der Schablone "pf_augment":

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

Angepasstes Profil erstellen

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

Parameter

-create
Erstellt ein Profil. (Erforderlich)

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

Verwenden Sie das folgende Format:

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

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":

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

- Mit der Schablone "pf_augment":

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

Eigenständiges Anwendungsserverprofil erstellen

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

Parameter

-create
Erstellt ein Profil. (Erforderlich)

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

Verwenden Sie das folgende Format:

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

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":

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

- Mit der Schablone "pf_augment":

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

Zellenprofil erstellen

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

Parameter

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

-create
Erstellt ein Profil. (Erforderlich)

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

Verwenden Sie das folgende Format:

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

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Geben Sie die folgenden Parameter für die Anwendungsserverschablone an:

-create
Erstellt ein Profil. (Erforderlich)

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

Verwenden Sie das folgende Format:

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

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":

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

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

- Mit der Schablone "pf_augment":

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

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

Deployment-Manager-Profil erweitern

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

Parameter

-augment
Erweitert ein vorhandenes Profil. (Erforderlich)

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

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

Verwenden Sie das folgende Format:

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

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":

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

- Mit der Schablone ″pf_augment″:

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

Angepasstes Profil erweitern

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

Parameter

-augment
Erweitert ein vorhandenes Profil. (Erforderlich)

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

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

Verwenden Sie das folgende Format:

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

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone ″xs_augment″:

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

- Mit der Schablone ″pf_augment″:

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

Eigenständiges Anwendungsserverprofil erweitern

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

Parameter

-augment
Erweitert ein vorhandenes Profil. (Erforderlich)

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

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

Verwenden Sie das folgende Format:

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

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone ″xs_augment″:

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

- Mit der Schablone ″pf_augment″:

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

Zellenprofil erweitern

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

Parameter

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

-augment
Erweitert ein vorhandenes Profil. (Erforderlich)

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

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

Verwenden Sie das folgende Format:

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

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Geben Sie die folgenden Parameter für das Anwendungsserverprofil an:

-augment
Erweitert ein vorhandenes Profil. (Erforderlich)

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

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

Verwenden Sie das folgende Format:

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

Schablonentyp steht für `xs_augment` oder `pf_augment`.

Beispiel

- Mit der Schablone "xs_augment":

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

- Mit der Schablone "pf_augment":

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

Erweiterung eines Profils aufheben

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

Parameter

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

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

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

Verwenden Sie das folgende Format:

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

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

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

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

Beispiel

- Mit der Schablone "xs_augment":

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

- Mit der Schablone "pf_augment":

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

Zugehörige Tasks

„Profile für WebSphere eXtreme Scale erstellen und erweitern" auf Seite 42
Nachdem Sie das Produkt installiert haben, erstellen Sie eindeutige Typen von Profilen und erweitern vorhandene Profile für WebSphere eXtreme Scale.

„Grafische Benutzerschnittstelle für die Erstellung von Profilen verwenden" auf Seite 43
Verwenden Sie die grafische Benutzerschnittstelle (GUI), die mit dem PMT-Plug-in (Profile Management Tool) bereitgestellt wird, um Profile für WebSphere eXtreme Scale zu erstellen. Ein Profil besteht aus einer Gruppe von Dateien, die die Laufzeitumgebung definieren.

„Grafische Benutzerschnittstelle für die Erweiterung von Profilen verwenden" auf Seite 44
Nach der Installation des Produkts können Sie ein vorhandenes Profil erweitern, um es mit WebSphere eXtreme Scale kompatibel zu machen.

Profile ohne Root-Rechte

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

Insgesamt betrachtet sind Benutzer ohne Root-Rechte hinsichtlich der Erstellung und Verwendung von Profilen in ihrer Umgebung eingeschränkt. Im PMT-Plug-in (Profile Management Tool) sind eindeutige Namen und Portwerte für Benutzer ohne Root-Rechte inaktiviert. Ein Benutzer ohne Root-Rechte muss die Standardfeldwerte für Profilnamen, Knotennamen, Zellennamen und Portzuordnungen in Profile Management Tool ändern. Daher sollte den Benutzern ohne Root-Rechte ein

Wertebereich für jedes dieser Felder zugewiesen werden. Sie können die Zuständigkeit für die Verwendung der richtigen Wertebereiche und die Wahrung der Integrität ihrer eigenen Definitionen an die Benutzer delegieren.

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

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

Weitere Einzelheiten finden Sie in den ausführlichen Informationen zum Erstellen von Profilen als Benutzer ohne Root-Rechte, in denen auch die Schritte zum Ausführen der vorherigen Task-Beispiele beschrieben sind. Sie finden diese Informationen im Information Center von WebSphere Application Server Network Deployment.

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

Wenn ein Benutzer ohne Root-Rechte jedoch ein Profil erweitert, das vom Installationsverantwortlichen erstellt wurde, muss der Benutzer ohne Root-Rechte die folgenden Dateien vor der Erweiterung nicht erstellen, weil sie bereits während der Profilerstellung erstellt wurden:

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

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

WebSphere eXtreme Scale unbeaufsichtigt installieren

Verwenden Sie eine vollständig qualifizierte Antwortdatei, die Sie an Ihre speziellen Anforderungen anpassen, oder übergeben Sie in der Befehlszeile Parameter, um WebSphere eXtreme Scale unbeaufsichtigt zu installieren.

Vorbereitungen

Stoppen Sie alle Prozesse, die in Ihrer Umgebung mit WebSphere Application Server oder WebSphere Application Server Network Deployment aktiv sind. Weitere Informationen zu den Befehlen stopManager, stopNode und stopServer finden Sie im Information-Center-Artikel Befehlszeilendienstprogramme.

Warum und wann dieser Vorgang ausgeführt wird

Bei einer unbeaufsichtigten Installation wird dasselbe Installationsprogramm verwendet, das auch bei der Installation über die grafische Benutzerschnittstelle verwendet wird. Anstatt jedoch eine Assistentenschnittstelle anzuzeigen, werden bei der unbeaufsichtigten Installation alle Antworten aus einer Datei, die Sie anpassen, oder aus Parametern gelesen, die Sie an die Befehlszeile übergeben. Die Datei "wxssetup.response.txt" ist eine Beispielantwortdatei, die für jede Option eine Beschreibung enthält.

1. Optional: Wenn Sie sich für die Installation von eXtreme Scale über eine Antwortdatei entscheiden, müssen Sie die Datei zuerst anpassen.

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

   a. Kopieren Sie die Antwortdatei von der Produkt-DVD auf Ihr Plattenlaufwerk.
   b. Öffnen Sie die Antwortdatei mit einem Texteditor Ihrer Wahl, und bearbeiten Sie sie. Sie müssen die folgenden Parameter angeben:
      - die Lizenzvereinbarung,
      - das Installationsverzeichnis.

      Tipp: Wenn Sie eXtreme Scale in einer Umgebung mit WebSphere Application Server installieren, verwendet das Installationsprogramm das Installationsverzeichnis, um festzustellen, wo die vorhandene Installation von WebSphere Application Server installiert ist. Wenn Sie die Installation auf einem Knoten durchführen, der mehrere Instanzen von WebSphere Application Server enthält, müssen Sie Ihre Position eindeutig definieren.
   c. Führen Sie das folgende Script aus, um die Installation zu starten.

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

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

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

Installationsparameter

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

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

Parameter

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

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

Beispielsyntax

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

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

Beispielsyntax

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

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

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

Beispielsyntax

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

-is:log *Pfadname/Dateiname*
Erstellt eine Protokolldatei, die die JVM-Suchen des Installationsprogramms beim Versuch, die GUI zu starten, enthält. Die Protokolldatei wird nicht erstellt, sofern dies nicht angegeben wird.

Beispielsyntax

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

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

Beispielsyntax

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

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

Beispielsyntax

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

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

Beispielsyntax

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

Zugehörige Tasks

„Eigenständige Version von WebSphere eXtreme Scale installieren" auf Seite 29
Sie können die eigenständige Version von WebSphere eXtreme Scale in einer Umgebung installieren, die weder WebSphere Application Server noch WebSphere Application Server Network Deployment enthält.

„WebSphere eXtreme Scale deinstallieren" auf Seite 58
Wenn Sie WebSphere eXtreme Scale aus Ihrer Umgebung entfernen möchten, können Sie dazu den Assistenten verwenden, oder Sie können das Produkt im unbeaufsichtigten Modus deinstallieren.

„WebSphere eXtreme Scale in WebSphere Application Server integrieren" auf Seite 31
Sie können WebSphere eXtreme Scale in einer Umgebung installieren, in der WebSphere Application Server oder WebSphere Application Server Network Deployment installiert ist. Sie können die vorhandenen Features von WebSphere Application Server oder WebSphere Application Server Network Deployment verwenden, um Ihre Anwendungen zu erweitern, indem Sie die Funktionalität von eXtreme Scale anwenden.

„Profile für WebSphere eXtreme Scale erstellen und erweitern" auf Seite 42
Nachdem Sie das Produkt installiert haben, erstellen Sie eindeutige Typen von Profilen und erweitern vorhandene Profile für WebSphere eXtreme Scale.

Update Installer zum Installieren von Wartungspaketen verwenden

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

Warum und wann dieser Vorgang ausgeführt wird

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

1. Stoppen Sie alle Prozesse, die in Ihrer Umgebung aktiv sind.
   - Weitere Informationen zum Stoppen aller Prozesse, die in der eigenständigen eXtreme-Scale-Umgebung ausgeführt werden, finden Sie im Abschnitt „Eigenständige eXtreme-Scale-Server stoppen" auf Seite 239.
   - Weitere Informationen zum Stoppen aller Prozesse, die in der Umgebung von WebSphere Application Server ausgeführt werden, finden Sie auf der Webseite Befehlszeilendienstprogramme.
2. Laden Sie die aktuelle Version von Update Installer herunter. Weitere Informationen finden Sie auf der Webseite Recommended fixes.
3. Installieren Sie Update Installer. Weitere Informationen finden Sie im Artikel Update Installer für WebSphere Software im Information Center von WebSphere Application Server.

4. Laden Sie die Wartungspakete, die Sie installieren möchten, in das Verzeichnis *UPDI-Stammverzeichnis*/maintenance herunter. Weitere Informationen finden Sie auf der Unterstützungssite.
5. Verwenden Sie Update Installer, um den vorläufigen Fix, das Fixpack oder Refresh-Pack zu installieren. Sie können das Wartungspaket installieren, indem Sie die grafische Benutzerschnittstelle verwenden oder Update Installer im unbeaufsichtigten Modus ausführen.

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

   - update.sh
   - update.bat

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

   - ./update.sh -silent -options *responsefile/Dateiname*
   - update.bat -silent -options *responsefile\Dateiname*

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

Angepassten Object Request Broker konfigurieren

Sie können eine angepasste Version des Object Request Broker (ORB) mit WebSphere eXtreme Scale verwenden, wenn Sie eigenständige Java-SE-Prozesse in Ihrer Umgebung ausführen.

Vorbereitungen

WebSphere eXtreme Scale und WebSphere Application Server stellen beide einen ORB bereit, der bereits für die Verwendung mit eXtreme Scale konfiguriert ist. Unter normalen Umständen müssen Sie den ORB nicht konfigurieren und auch keinen anderen ORB verwenden.

Warum und wann dieser Vorgang ausgeführt wird

WebSphere eXtreme Scale verwendet den Object Request Broker (ORB), um die Kommunikation zwischen Prozessen zu ermöglichen. Ein ORB wird mit eXtreme Scale und WebSphere Application Server bereitgestellt. Wenn Sie ein IBM Developer Kit oder ein SDK verwenden, das mit WebSphere Application Server bereitgestellt wird, ist der ORB in der JRE enthalten.

Sie können den ORB, der mit eXtreme Scale bereitgestellt wird, den ORB, der mit IBM SDK bereitgestellt wird, oder den ORB, der mit WebSphere Application Server bereitgestellt wird, verwenden. Stellen Sie sicher, dass alle Probleme, die bei der Verwendung von ORBs unabhängiger Softwareanbieter auftreten, mit dem IBM ORB und einer kompatiblen JRE reproduzierbar sind, bevor Sie Unterstützung anfordern. eXtreme Scale bietet keine Unterstützung für den ORB, der mit dem Sun Microsystems Java Development Kit (JDK) bereitgestellt wird. Obwohl eXtreme Scale Developer-Kits von den meisten Anbietern unterstützt, wird empfohlen, den ORB zu verwenden, der mit eXtreme Scale bereitgestellt wird.

- Wenn Ihre Umgebung ein SDK der Version 5 oder hoher enthält, aktualisieren Sie die Scripts, die den Java-Befehl starten, indem Sie ein anderes Verzeichnis angeben.

1. Kopieren Sie die angepasste Datei `ibmorb.jar` und die Datei `ibmorbapi.jar` in ein leeres Verzeichnis.

– Führen Sie den folgende Schritt aus, wenn Sie Produktscripts in einer eigenständigen eXtreme-Scale-Umgebung verwenden.

1. Ändern Sie den Pfad für die Variable OBJECTGRID_ENDORSED_DIRS in der Datei "setupCmdLine" so, dass sie auf das angepasste ORB-Verzeichnis verweist. Speichern Sie Ihre Änderungen.

   Editieren Sie die Datei *ObjectGrid-Stammverzeichnis*`/bin/setupCmdLine.sh`.

   Editieren Sie die Datei *ObjectGrid-Stammverzeichnis*`\bin\setupCmdLine.bat`.

– Führen Sie den folgenden Schritt aus, wenn Sie Produktscripts in einer Umgebung mit WebSphere Application Server verwenden:

1. Fügen Sie dem Script "startOgServer" die folgende Systemeigenschaft und die folgenden Parameter hinzu:

   `-jvmArgs –Djava.endorsed.dirs=`*`angepasstes_ORB-Verzeichnis`*

– Führen Sie den folgenden Schritt aus, wenn Sie ein angepasstes Script zum Starten eines Clientanwendungsprozesses oder Serverprozesses verwenden möchten:

1. Fügen Sie dem angepassten Script die folgende Systemeigenschaft hinzu:

   `-Djava.endorsed.dirs=`*`angepasstes_ORB-Verzeichnis`*

- Wenn Ihre Umgebung ein SDK der Version 1.4.2 enthält, integrieren Sie den IBM ORB im angegebenen SDK.

1. Laden Sie den ORB aus einem IBM SDK herunter, und extrahieren Sie ihn. Wenn kein IBM SDK für Ihre Plattform verfügbar ist, laden Sie das IBM Developer Kit für Linux, Java Technology Edition herunter, und extrahieren Sie es. Weitere Informationen finden Sie auf der Webseite IBM Developer Kits.
2. Kopieren Sie die Dateien `java/jre/lib/ibmorb.jar` und `java/jre/lib/ibmorbapi.jar` in das Verzeichnis `java/jre/lib/ext` des Ziel-SDK.
3. Erstellen oder editieren Sie die Datei `orb.properties` im Verzeichnis `java/jre/lib` des SDK. Fügen Sie die folgenden Eigenschaften hinzu, bzw. stellen Sie sicher, dass die folgenden Eigenschaften in der Datei vorhanden sind:

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

   Beschreibungen der Eigenschaften und Einstellungen finden Sie im Abschnitt „ORB-Eigenschaftendatei" auf Seite 207.
4. Laden Sie Xerces2 Java 2.9 herunter. Weitere Informationen finden Sie auf der Webseite The Apache Xerces Project - Downloads.
5. Kopieren Sie die Dateien `xercesImpl.jar` und `xml-apis.jar` in das Verzeichnis `lib/ext`.

Ergebnisse

Sie können den angepassten ORB mit eXtreme Scale verwenden, wenn Sie eigenständige Java-SE-Client- und -Serverprozesse verwenden.

Zugehörige Verweise

„ORB-Eigenschaftendatei” auf Seite 207
Die Datei `orb.properties` wird für die Übergabe der vom Object Request Broker (ORB) verwendeten Eigenschaften verwendet, um das Transportverhalten des Grids zu ändern.

Angepasste Eigenschaften des Object Request Broker (ORB)

WebSphere eXtreme Scale deinstallieren

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

Vorbereitungen

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

1. Stoppen Sie alle Prozesse, in denen eXtreme Scale ausgeführt wird. Andernfalls schlägt die Deinstallation fehl.
   - Wenn Sie die eigenständige Version von eXtreme Scale installiert haben, lesen Sie den Abschnitt „Eigenständige eXtreme-Scale-Server stoppen” auf Seite 239.
   - Wenn Sie eXtreme Scale in einer vorhandenen Installation von WebSphere Application Server installiert haben, finden Sie auf der Webseite Befehlszeilendienstprogramme weitere Informationen zum Stoppen von Prozessen von WebSphere Application Server.
2. Führen Sie das folgende Script aus:
   - `Installationsstammverzeichnis/uninstall_wxs/uninstall`.
   - `Installationsstammverzeichnis\uninstall_wxs\uninstall.exe`

Ergebnisse

Sie haben eXtreme Scale aus Ihrer Umgebung entfernt.

Zugehörige Verweise

„Installationsparameter” auf Seite 53
Geben Sie Parameter in der Befehlszeile an, um Ihre Produktinstallation anzupassen und zu konfigurieren.

Kapitel 5. WebSphere eXtreme Scale for z/OS anpassen

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

Vorbereitungen

- Vergewissern Sie sich, dass Ihr System die aktuelle Version von WebSphere Application Server Network Deployment enthält:
  - Wenn Sie mit Version 6.1 arbeiten, muss Ihr System mindestens Fix Pack 27 enthalten. Weitere Informationen finden Sie im Artikel Anwendungsserverumgebung der Version 6.1 installieren.
  - Wenn Sie mit Version 7.0 arbeiten, muss Ihr System mindestens Fix Pack 3 enthalten. Weitere Informationen finden Sie im Artikel Anwendungsserverumgebung der Version 7.0 installieren.
- Installieren Sie WebSphere eXtreme Scale for z/OS. Weitere Informationen finden Sie unter WebSphere eXtreme Scale Program Directory auf der Bibliotheksseite.

Warum und wann dieser Vorgang ausgeführt wird

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

- „WebSphere Customization Tools installieren"
- „Anpassungsdefinitionen generieren" auf Seite 60
- „Angepasste Jobs hochladen und ausführen" auf Seite 61

WebSphere Customization Tools installieren

Installieren Sie WebSphere Customization Tools Version 7.0.0.3 oder höher, um Ihre Umgebung von WebSphere eXtreme Scale for z/OS anzupassen.

Vorbereitungen

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

Warum und wann dieser Vorgang ausgeführt wird

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

1. Verwenden Sie FTP, um die Erweiterungsdateien `xs.wct` und `xspf.wct` von Ihrem z/OS-System auf die Workstation zu kopieren, auf der Sie WebSphere Customization Tools installieren. Die Erweiterungsdateien befinden sich im Verzeichnis `/usr/lpp/zWebSphereXS/util/V7R0/WCT` auf Ihrem z/OS-System.
2. Laden Sie WebSphere Customization Tools Version 7.0.0.3 oder höher von der entsprechenden Website herunter, und installieren Sie das Produkt:
   - WebSphere Customization Tools for Windows
   - WebSphere Customization Tools for Linux

3. Laden Sie die Datei `xs.wct` in die Anwendung WebSphere Customization Tools hoch.
   a. Starten Sie die Anwendung WebSphere Customization Tools auf Ihrer Workstation.
   b. Klicken Sie auf **Help → Software Updates → Install Extension**.
   c. Klicken Sie in der Anzeige "WebSphere Customization Tools Extension Locations" auf **Install new extension location**.
   d. Klicken Sie in der Anzeige "Source Archive File" auf **Browse**, navigieren Sie zu dem Verzeichnis, in das Sie die Datei `xs.wct` in Schritt 1 kopiert haben, und klicken Sie auf **Open**.
   e. Klicken Sie in der Anzeige "Summary" auf **Next**.

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

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

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

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

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

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

Nächste Maßnahme

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

Anpassungsdefinitionen generieren

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

Vorbereitungen

Installieren Sie WebSphere Customization Tools, und laden Sie die Erweiterungsdateien `xs.wct` und `xspf.wct` hoch. Weitere Informationen finden Sie im Abschnitt „WebSphere Customization Tools installieren" auf Seite 59.

Warum und wann dieser Vorgang ausgeführt wird

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

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

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

Nächste Maßnahme

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

Angepasste Jobs hochladen und ausführen

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

Vorbereitungen

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

„Anpassungsdefinitionen generieren” auf Seite 60.

Warum und wann dieser Vorgang ausgeführt wird

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

1. Laden Sie die angepassten Jobs hoch. Wählen Sie auf der Registerkarte **Anpassungsdefinitionen** die Jobs aus, die Sie hochladen möchten, und klicken Sie anschließend auf **Verarbeiten**.
2. Laden Sie die Jobs auf den FTP-Server auf Ihrem z/OS-System hoch. Geben Sie die erforderlichen Informationen in der Anzeige **Anpassungsdefinition hochladen** an.
3. Klicken Sie auf **Fertig stellen**.
4. Führen Sie die angepassten Jobs aus. Klicken Sie auf das Register **Anpassungsanweisungen**, und folgen Sie den Anpassungsanweisungen für jeden Job.

Kapitel 6. WebSphere eXtreme Scale konfigurieren

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

Lokales speicherinternes ObjectGrid konfigurieren

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

Warum und wann dieser Vorgang ausgeführt wird

Es folgt ein einfaches XML-Beispiel, die Datei `companyGrid.xml`. Die ersten Zeilen der Datei enthalten den erforderlichen Header, den jede ObjectGrid-XML-Datei enthalten muss. Die Datei definiert das ObjectGrid ″CompanyGrid″ mit den BackingMaps ″Customer″, ″Item″, ″OrderLine″ und ″Order″. eine XML-Implementierungsrichtliniendatei wird während des Starts an einen Container von eXtreme Scale übergeben.

companyGrid.xml file

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

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

</objectGridConfig>
```

Übergeben Sie die XML-Datei an eine der createObjectGrid-Methoden in der Schnittstelle ″ObjectGridManager. Der folgende Mustercode validiert die Datei `companyGrid.xml` anhand des XML-Schemas und erstellt das ObjectGrid ″CompanyGrid″. Die neu erstellte ObjectGrid-Instanz wird nicht zwischengespeichert.

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

Als Alternative zur Verwendung von XML können ObjectGrid-Objekte auch programmgesteuert erstellt werden. Der folgende Mustercode kann an Stelle der vorherigen XML und des vorherigen Codes verwendet werden:

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

eXtreme Scale hat viele anpassbare Plug-ins und Attribute. Eine vollständige Beschreibung der ObjectGrid-XML-Datei finden Sie in den Referenzinformationen zur Konfiguration von eXtreme Scale.

Schnittstelle "ObjectGrid"

Mit den folgenden Methoden können Sie mit einer ObjectGrid-Instanz interagieren.

Einführung

Erstellen und initialisieren

Die erforderlichen Schritte zum Erstellen einer ObjectGrid-Instanz finden Sie im Abschnitt zur Schnittstelle "ObjectGridManager". Es gibt zwei Methoden zum Erstellen einer ObjectGrid-Instanz. Sie können eine ObjectGrid-Instanz programmgesteuert oder über XML-Konfigurationsdateien erstellen. Weitere Informationen finden Sie in der API-Dokumentation.

Get-, Set- und Factory-Methoden

Alle Set-Methoden müssen vor der Initialisierung der ObjectGrid-Instanz aufgerufen werden. Wenn Sie eine Set-Methode nach dem Aufruf der Methode "initialize" aufrufen, wird eine Ausnahme des Typs "java.lang.IllegalStateException" ausgegeben. Jede getSession-Methode der Schnittstelle "ObjectGrid" ruft implizit auch die Methode "initialize" auf. Deshalb müssen die Set-Methoden vor dem Aufruf der getSession-Methoden aufgerufen werden. Die einzige Ausnahme von dieser Regel sind das Festlegen, Hinzufügen und Entfernen von ObjectGridEventListener-Objekten. Diese Objekte können nach Abschluss des Initialisierungsprozesses verarbeitet werden.

Die ObjectGrid-Schnittstelle enthält die folgenden Hauptmethoden.

Tabelle 5. Methoden der Schnittstelle "ObjectGrid"

Methode	Beschreibung
BackingMap defineMap(String name);	defineMap: Diese Methode ist eine Factory-Methode, mit der eine eindeutig benannte BackingMap definiert werden kann. Weitere Informationen zu BackingMaps finden Sie in der Beschreibung der Schnittstelle "BackingMap".
BackingMap getMap(String name);	getMap: Gibt eine BackingMap zurück, die zuvor durch den Aufruf von "defineMap" definiert wurde. Mit dieser Methode können Sie die BackingMap konfigurieren, wenn diese noch nicht durch XML-Konfiguration konfiguriert wurde.
BackingMap createMap(String name);	createMap: Erstellt eine BackingMap, stellt sie aber nicht zur Verwendung durch das ObjectGrid in den Cache. Verwenden Sie diese Methode mit der Methode "setMaps(List)" der Schnittstelle "ObjectGrid", die BackingMaps zur Verwendung mit diesem ObjectGrid zwischenspeichert. Verwenden Sie diese Methoden, wenn Sie ein ObjectGrid mit dem Spring-Framework konfigurieren.
void setMaps(List mapList);	setMaps: Löscht alle BackingMaps, die zuvor in diesem ObjectGrid definiert wurden, und ersetzt sie durch die bereitgestellte Liste mit BackingMaps.
public Session getSession() throws ObjectGridException, TransactionCallbackException;	getSession: Gibt ein Session-Objekt zurück, das begin-, commit- und rollback-Funktionen für eine Arbeitseinheit bereitstellt. Weitere Informationen zu Session-Objekten finden Sie in der Beschreibung der Schnittstelle "Session".
Session getSession(CredentialGenerator cg);	getSession(CredentialGenerator cg): Ruft ein Session-Objekt mit einem CredentialGenerator-Objekt ab. Diese Methode kann nur vom ObjectGrid-Client in einer Client/Server-Umgebung aufgerufen werden.
Session getSession(Subject subject);	getSession(Subject subject): Lässt die Verwendung eines speziellen Subject-Objekts an Stelle des einen in ObjectGrid konfigurierten zu, um ein Session-Objekt abzurufen.

Tabelle 5. Methoden der Schnittstelle "ObjectGrid" (Forts.)

Methode	Beschreibung
void initialize() throws ObjectGridException;	initialize: Das ObjectGrid wird initialisiert und steht für die allgemeine Verwendung zur Verfügung. Diese Methode wird beim Aufruf der Methode Methode "getSession" implizit aufgerufen, falls das ObjectGrid noch nicht initialisiert ist.
void destroy();	destroy: Das Framework wird zerlegt und kann nach dem Aufruf dieser Methode nicht mehr verwendet werden.
void setTxTimeout(int timeout);	setTxTimeout: Verwenden Sie diese Methode, um die Zeit (in Sekunden) festzulegen, in der eine Transaktion, die in einer von dieser ObjectGrid-Instanz erstellten Sitzung gestartet wird, ausgeführt werden muss. Wenn eine Transaktion nicht in der angegebenen Zeit beendet wird, wird die Sitzung, in der die Transaktion gestartet wurde, mit "timed out" (Zeitlimitüberschreitung) markiert. Dies führt dazu, dass die nächste ObjectMap-Methode, die von der Sitzung, die das zulässige Zeitlimit überschritten hat, aufgerufen wird, eine Ausnahme des Typs auslöst. Die Sitzung wird mit "rollback only" (Nur Rollback) markiert, was dazu führt, dass die Transaktion auch dann rückgängig gemacht wird, wenn die Anwendung die Methode "commit" an Stelle der Methode "rollback" aufruft, nachdem die Ausnahme des Typs "TransactionTimeoutException" von der Anwendung abgefangen wurde. Ein Zeitlimit von 0 zeigt an, dass der Transaktion kein Zeitlimit für die Ausführung gesetzt wird. Die Transaktion überschreitet nie ein Zeitlimit, wenn 0 als Zeitlimitwert angegeben wird. Wenn diese Methode nicht aufgerufen wird, wird für jedes Session-Objekt, das von der Methode "getSession" dieser Schnittstelle zurückgegeben wird, standardmäßig ein Zeitlimitwert von 0 festgelegt. Eine Anwendung kann die Transaktionszeitlimiteinstellung für jedes Session-Objekt überschreiben, indem sie die Methode "setTransactionTimeout" der Schnittstelle "com.ibm.websphere.objectgrid.Session" verwendet. Sie können das Transaktionszeitlimit für verteilte Instanzen auch mit der Datei `objectGrid.xml` konfigurieren.
int getTxTimeout();	getTxTimeout: Gibt den Transaktionszeitlimitwert in Sekunden zurück. Diese Methode gibt den Wert zurück, der als Zeitlimitparameter in der Methode "setTxTimeout" übergeben wurde. Wenn die Methode "setTxTimeout" nicht aufgerufen wurde, gibt die Methode "0" zurück, um anzuzeigen, dass für die Transaktion kein Ausführungszeitlimit gesetzt ist.
public int getObjectGridType();	Gibt den Typ des ObjectGrids zurück. Der Rückgabewert ist eine der in dieser Schnittstelle deklarierten Konstanten: LOCAL, SERVER oder CLIENT.
public void registerEntities(Class[] entities);	Registriert eine oder mehreren Entitäten auf der Basis der Metadaten der Klasse. Die Entitätsregistrierung muss vor der ObjectGrid-Initialisierung stattfinden, zum eine Entität an eine BackingMap und alle definierten Indizes zu binden. Diese Methode kann mehrfach aufgerufen werden.
public void registerEntities(URL entityXML);	Registriert eine oder mehrere Entitäten über eine XML-Entitätsdatei. Die Entitätsregistrierung muss vor der ObjectGrid-Initialisierung stattfinden, zum eine Entität an eine BackingMap und alle definierten Indizes zu binden. Diese Methode kann mehrfach aufgerufen werden.
void setQueryConfig(QueryConfig queryConfig);	Legt das QueryConfig-Objekt für dieses ObjectGrid fest. Ein QueryConfig-Objekt stellt Abfragekonfigurationen für die Ausführung von Objektabfragen für die Maps in diesem ObjectGrid bereit.
//Keywords	
void associateKeyword(Serializable parent, Serializable child);	Veraltet: Verwenden Sie Index- oder Abfragefunktionen, um Objekte mit bestimmten Attributen abzurufen. Die Methode "associateKeyword" ist ein flexibler Mechanismus für das Ungültigmachen von Einträgen auf der Basis von Schlüsselwörtern. Diese Methode verknüpft die beiden Schlüsselwörter zu einer gerichteten Beziehung. Wenn der übergeordnete Eintrag ungültig gemacht wird, wird auch der untergeordnete Eintrag ungültig gemacht. Das Ungültigmachen des untergeordneten Eintrags hat keine Auswirkung auf den übergeordneten Eintrag.
//Sicherheit	
void setSecurityEnabled()	setSecurityEnabled: Aktiviert die Sicherheit. Die Sicherheit ist standardmäßig inaktiviert.
void setPermissionCheckPeriod(long period);	setPermissionCheckPeriod: Diese Methode akzeptiert einen einzigen Parameter, der angibt, wie oft die Berechtigung geprüft wird, die verwendet wird, um einen Clientzugriff zuzulassen. Wenn der Parameter 0 ist, weisen alle Methoden den Berechtigungsmechanismus (JAAS-Berechtigung oder angepasste Berechtigung) an, zu prüfen, ob das aktuelle Subject-Objekt berechtigt ist. Diese Strategie kann je nach Berechtigungsimplementierung zu Leistungsproblemen führen. Dieser Typ von Berechtigung ist jedoch verfügbar, wenn es erforderlich ist. Wenn der Parameter einen Wert kleiner als 0 hat, gibt dieser die Anzahl an Millisekunden an, für die eine Gruppe von Berechtigungen zwischengespeichert wird, bevor sie vom Berechtigungsmechanismus aktualisiert wird. Dieser Parameter bietet eine sehr viel bessere Leistung, aber wenn die Back-End-Berechtigungen in dieser Zeit geändert werden, ist es möglich, dass das ObjectGrid den Zugriff zulässt oder verhindert, auch wenn der Back-End-Sicherheitsprovider geändert wurde.

Tabelle 5. Methoden der Schnittstelle ″ObjectGrid″ (Forts.)

Methode	Beschreibung
void setAuthorizationMechanism(int authMechanism);	setAuthorizationMechanism: Legt den Berechtigungsmechanismus fest. Die Standardeinstellung ist SecurityConstants.JAAS_AUTHORIZATION.
setMapAuthorization(MapAuthorization ma);	Veraltet: Verwenden Sie die Methode ″setObjectGridAuthorization (ObjectGridAuthorization)″ an Stelle der Integration angepasster Berechtigungen. setMapAuthorization: Legt das MapAuthorization-Plug-in für diese ObjectGrid-Instanz fest. Dieses Plug-in kann verwendet werden, um ObjectMap- oder JavaMap-Zugriffe auf die Principals zu berechtigen, die im Subject-Objekt enthalten sind. Eine typische Implementierung dieses Plug-ins ist der Abruf der Principals aus dem Subject-Objekt mit anschließender dahingehender Prüfung, ob die angegebenen Berechtigungen den Principals erteilt wurden oder nicht.
setSubjectSource(SubjectSource ss);	setSubjectSource: Legt das SubjectSource-Plug-in fest. Dieses Plug-in kann verwendet werden, um ein Subject-Objekt abzurufen, das den ObjectGrid-Client darstellt. Dieses Subject-Objekt wird für die ObjectGrid-Berechtigung verwendet. Die Methode ″SubjectSource.getSubject″ wird von der ObjectGrid-Laufzeitumgebung aufgerufen, wenn die Methode ″ObjectGrid.getSession″ zum Abrufen einer Sitzung verwendet wird und die Sicherheit aktiviert ist. Dieses Plug-in ist hilfreich für einen bereits authentifizierten Client. Es kann das authentifizierte Subject-Objekt abrufen und anschließend an die ObjectGrid-Instanz übergeben. Eine weitere Authentifizierung ist nicht erforderlich.
setSubjectValidation(SubjectValidation sv);	setSubjectValidation: Legt das SubjectValidation-Plug-in für diese ObjectGrid-Instanz fest. Dieses Plug-in kann verwendet werden, um zu prüfen, ob ein javax.security.auth.Subject-Subject-Objekt, das an die ObjectGrid-Instanz übergeben wird, ein gültiges Subject-Objekt ist, das nicht manipuliert wurde. Für die Implementierung dieses Plug-ins wird die Unterstützung des Subject-Objekterstellers benötigt, da nur der Ersteller weiß, ob das Subject-Objekt manipuliert wurde. Es kann jedoch vorkommen, dass ein Subject-Ersteller nicht weiß, ob das Subject-Objekt manipuliert wurde. In diesem Fall sollte dieses Plug-in nicht verwendet werden.
void setObjectGridAuthorization(ObjectGridAuthorization ogAuthorization);	Legt die ObjectGridAuthorization für diese ObjectGrid-Instanz fest. Bei der Übergabe von null an diese Methode wird ein zuvor definiertes ObjectGridAuthorization-Objekt aus einem früheren Aufruf dieser Methode entfernt. Außerdem wird damit angezeigt, dass dieses <code>ObjectGrid</code> keinem ObjectGridAuthorization-Objekt zugeordnet ist. Diese Methode sollte nur verwendet werden, wenn die ObjectGrid-Sicherheit aktiviert ist. Wenn die ObjectGrid-Sicherheit inaktiviert ist, wird das bereitgestellte ObjectGridAuthorization-Objekt nicht verwendet. Ein ObjectGridAuthorization -Plug-in kann verwendet werden, um den Zugriff auf das ObjectGrid und die Maps zu berechtigen. Ab Extended Deployment Version 6.1 ist setMapAuthorization veraltet und setObjectGridAuthorization die empfohlene Methode. Wenn jedoch das MapAuthorization-Plug-in und das ObjectGridAuthorization-Plug-in verwendet werden, verwendet ObjectGrid das bereitgestellte MapAuthorization-Plug-in, um Map-Zugriffe zu berechtigen, obwohl es veraltet ist.

Schnittstelle ″ObjectGrid″: Plug-ins

Die Schnittstelle ″ObjectGrid″ hat mehrere optionale Plug-in-Punkte für erweiterbare Interaktionen.

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

- ObjectGridEventListener: Eine ObjectGridEventListener-Schnittstelle wird verwendet, um Benachrichtigungen über wichtige Ereignisse zu empfangen, die im ObjectGrid eintreten. Zu diesen Ereignissen gehören die Initialisierung von ObjectGrid, der Beginn einer Transaktion, das Ende einer Transaktion und das Löschen eines ObjectGrids. Um diese Ergebnisse zu empfangen, erstellen Sie eine Klasse, die die Schnittstelle ″ObjectGridEventListener″ implementiert, und fügen Sie sie dem ObjectGrid hinzu. Diese Listener werden jedem einzelnen Session-Objekt zugeordnet. Weitere Informationen finden Sie in der Beschreibung von Listenern und in der Beschreibung der Schnittstelle ″Session″.
- TransactionCallback: Eine TransactionCallback-Listener-Schnittstelle ermöglicht das Senden von Transaktionsereignissen, wie z. B. begin-, commit- und rollback-

Signalen, an diese Schnittstelle. Gewöhnlich wird eine TransactionCallback-Listener-Schnittstelle zusammen mit einem Loader verwendet. Weitere Informationen finden Sie in der Beschreibung des TransactionCallback-Plug-ins und der Beschreibung von Loadern. Diese Ereignisse können anschließend verwendet werden, um Transaktionen mit einer externen Ressource oder in mehreren Loadern zu koordinieren.

- reserveSlot: Ermöglicht Plug-ins in diesem ObjectGrid, Slots für Objektinstanzen zu reservieren, die Slots haben, wie z. B. TxID.
- SubjectValidation. Wenn die Sicherheit aktiviert ist, kann dieses Plug-in verwendet werden, um eine javax.security.auth.Subject-Klasse zu validieren, die an das ObjectGrid übergeben wird.
- MapAuthorization. Wenn die Sicherheit aktiviert ist, kann dieses Plug-in verwendet werden, um ObjectMap-Zugriffe auf die Principals zu validieren, die vom Subject-Objekt dargestellt werden.
- SubjectSource: Wenn die Sicherheit aktiviert ist, kann dieses Plug-in verwendet werden, um ein Subject-Objekt abzurufen, das ObjectGrid-Client darstellt. Dieses Subject-Objekt wird anschließend für die ObjectGrid-Berechtigung verwendet.

Weitere Informationen zu Plug-ins finden Sie in der Einführung in Plug-ins im *Programmierhandbuch*.

Schnittstelle "BackingMap"

Jede ObjectGrid-Instanz enthält eine Sammlung von BackingMap-Objekten. Verwenden Sie die Methode "defineMap" oder die Methode "createMap" der Schnittstelle "ObjectGrid", um jede BackingMap zu benennen und einer ObjectGrid-Instanz hinzuzufügen. Diese Methoden geben eine BackingMap-Instanz zurück, die anschließend zum Definieren des Verhaltens einer einzelnen Map verwendet wird.

Schnittstelle "Session"

Die Schnittstelle "Session" wird verwendet, um eine Transaktion zu starten und das ObjectMap- bzw. JavaMap-Objekt abzurufen, das für die transaktionsorientierte Interaktion zwischen einer Anwendung und einem BackingMap-Objekt erforderlich ist. Die Transaktionsänderungen werden jedoch erst dann auf das BackingMap-Objekt angewendet, wenn die Transaktion festgeschrieben wird. Sie können sich eine BackingMap als Speichercache festgeschriebener Daten für eine einzelne Map vorstellen. Weitere Informationen finden Sie in den Informationen zur Verwendung von Session-Objekten für den Zugriff auf Daten im *Programmierhandbuch*.

Die Schnittstelle "BackingMap" stellt Methoden für die Festlegung von BackingMap-Attributen bereit. Einige der set-Methoden unterstützen die Erweiterbarkeit einer BackingMap über mehrere angepasste Plug-ins. In der folgenden Liste sind die set-Methoden für die Festlegung von Attributen und die Bereitstellung angepasster Plug-in-Unterstützung aufgeführt:

```
// Für die Festlegung von BackingMap-Attributen.
public void setReadOnly(boolean readOnlyEnabled);
public void setNullValuesSupported(boolean nullValuesSupported);
public void setLockStrategy( LockStrategy lockStrategy );
public void setCopyMode(CopyMode mode, Class valueInterface);
public void setCopyKey(boolean b);
public void setNumberOfBuckets(int numBuckets);
public void setNumberOfLockBuckets(int numBuckets);
public void setLockTimeout(int seconds);
public void setTimeToLive(int seconds);
public void setTtlEvictorType(TTLType type);
public void setEvictionTriggers(String evictionTriggers);

// Für die Festlegung eines optionalen angepassten Plug-ins, das von der Anwendung bereitgestellt wird.
public abstract void setObjectTransformer(ObjectTransformer t);
public abstract void setOptimisticCallback(OptimisticCallback checker);
```

```
public abstract void setLoader(Loader loader);
public abstract void setPreloadMode(boolean async);
public abstract void setEvictor(Evictor e);
public void setMapEventListeners( List /*MapEventListener*/ eventListenerList );
public void addMapEventListener(MapEventListener eventListener );
public void removeMapEventListener(MapEventListener eventListener );
public void addMapIndexPlugin(MapIndexPlugin index);
public void setMapIndexPlugins(List /\* MapIndexPlugin \*/ indexList );
public void createDynamicIndex(String name, boolean isRangeIndex,
String attributeName, DynamicIndexCallback cb);
public void createDynamicIndex(MapIndexPlugin index, DynamicIndexCallback cb);
public void removeDynamicIndex(String name);
```

Für jede der aufgelisteten set-Methoden gibt es eine entsprechende get-Methode.

BackingMap-Attribute

Jede BackingMap hat die folgenden Attribute, die gesetzt werden können, um das Verhalten der BackingMap zu ändern bzw. zu steuern:

- ReadOnly: Dieses Attribut gibt an, ob die Map eine schreibgeschützte Map oder eine Map mit Lese- und Schreibzugriff ist. Wenn dieses Attribut für die Map nicht gesetzt wird, ist die Map standardmäßig eine Map mit Lese- und Schreibzugriff. Wenn eine BackingMap schreibgeschützt ist, optimiert ObjectGrid die Leistung von Leseoperationen nur, wenn dies möglich ist.
- NullValuesSupported: Dieses Attribut gibt an, ob ein Nullwert in der Map gespeichert werden kann. Wenn dieses Attribut nicht gesetzt wird, unterstützt die Map keine Nullwerte. Wenn Nullwerte von der Map unterstützt werden, kann eine get-Operation, die null zurückgibt, bedeuten, dass der Wert null ist oder dass die Map den mit der get-Operation angegebenen Schlüssel nicht enthält.
- LockStrategy: Dieses Attribut bestimmt, ob ein Sperrenmanager von der BackingMap verwendet wird. Wenn ein Sperrenmanager verwendet wird, wird das Attribut "LockStrategy" verwendet, um anzuzeigen, ob ein optimistischer oder pessimistischer Sperransatz für das Sperren der Map-Einträge verwendet wird. Wenn dieses Attribut nicht gesetzt wird, wird die optimistische Sperrstrategie verwendet. Weitere Einzelheiten zu den unterstützten Sperrstrategien finden Sie im Abschnitt "Sperren".
- CopyMode: Dieses Attribut bestimmt, ob eine Kopie eines Wertobjekts von der BackingMap erstellt wird, wenn ein Wert aus der Map gelesen oder in der Map im Festschreibungszyklus einer Transaktion gespeichert wird. Es werden verschiedene Kopiermodi unterstützt, damit die Anwendung zwischen Leistung und Datenintegrität abwägen kann. Wenn dieses Attribut nicht gesetzt wird, wird der Kopiermodus COPY_ON_READ_AND_COMMIT verwendet. Dieser Kopiermodus bietet zwar nicht die beste Leistung, aber den größten Schutz vor Datenintegritätsproblemen. Wenn die BackingMap einer Entität der API "EntityManager" zugeordnet ist, hat die CopyMode-Einstellung keine Wirkung, sofern das Attribut nicht den Wert COPY_TO_BYTES hat. Jeder andere CopyMode-Einstellung wird automatisch auf NO_COPY gesetzt. Zum Überschreiben der CopyMode-Einstellung für eine Entitäts-Map verwenden Sie die Methode "ObjectMap.setCopyMode". Weitere Informationen zu den Kopiermodi finden Sie in den Informationen zu den bewährten Verfahren für die Methode "CopyMode" im *Programmierhandbuch*.
- CopyKey: Dieses Attribut bestimmt, ob die BackingMap eine Kopie eines Schlüsselobjekts erstellt, wenn ein Eintrag in der Map erstellt wird. Standardmäßig wird keine Kopie von Schlüsselobjekten erstellt, weil die Schlüssel normalerweise unveränderliche Objekte sind.
- NumberOfBuckets: Dieses Attribut gibt die Anzahl der von der BackingMap zu verwendenden Hash-Buckets an. Die BackingMap-Implementierung verwendet eine Hash-Tabelle für ihre Implementierung. Wenn sehr viele Einträge in der

BackingMap enthalten sind, kann mit einer höheren Bucket-Anzahl eine bessere Leistung erzielt werden. Die Anzahl der Schlüssel, die dasselbe Bucket haben, nimmt ab, je höher die Anzahl der Buckets wird. Mehr Buckets bedeuten auch mehr gemeinsame Zugriffe. Dieses Attribut ist hilfreich für die Feinabstimmung der Leistung. Es wird ein nicht definierter Standardwert verwendet, wenn die Anwendung das Attribut ″NumberOfBuckets″ nicht definiert.

- NumberOfLockBuckets: Dieses Attribut gibt die Anzahl der Sperr-Buckets an, die der Sperrenmanager für diese BackingMap verwenden soll. Wenn das Attribut ″LockStrategy″ auf OPTIMISTIC oder PESSIMISTIC gesetzt wird, wird ein Sperrenmanager für die BackingMap erstellt. Der Sperrenmanager verwendet eine Hash-Tabelle, um die Einträge zu verfolgen, die von einer oder mehreren Transaktionen gesperrt werden. Wenn sehr viele Einträge in der Hash-Tabelle enthalten sind, kann mit einer höheren Anzahl an Sperr-Buckets eine bessere Leistung erzielt werden, weil die Anzahl kollidierender Schlüssel im selben Bucket mit zunehmender Bucket-Anzahl sinkt. Mehr Sperr-Buckets bedeuten auch mehr gemeinsame Zugriffe. Wenn das Attribut ″LockStrategy″ auf ″NONE″ gesetzt wird, wird kein Sperrenmanager von der BackingMap verwendet. In diesem Fall hat die Definition des Attributs ″numberOfLockBuckets″ keine Wirkung. Wenn dieses Attribut nicht gesetzt wird, wird ein nicht definierter Wert verwendet.
- LockTimeout: Dieses Attribut wird verwendet, wenn die BackingMap einen Sperrenmanager verwendet. Die BackingMap verwendet einen Sperrenmanager, wenn das Attribut ″LockStrategy″ auf OPTIMISTIC oder PESSIMISTIC gesetzt ist. Der Attributwert wird in Sekunden angegeben und bestimmt, wie lange der Sperrenmanager auf die Erteilung einer Sperre wartet. Wenn dieses Attribut nicht gesetzt wird, werden 15 Sekunden als Wert für ″LockTimeout″ verwendet. Einzelheiten zu den Ausnahmen für das Wartezeitlimit für Sperren finden Sie in der Beschreibung der pessimistischen Sperrstrategie.
- TtlEvictorType: Jede BackingMap hat ein integriertes TTL-Bereinigungsprogramm (Time-to-Live, Lebensdauer), das einen zeitbasierten Algorithmus verwendet, um die zu entfernenden Map-Einträge zu bestimmen. Standardmäßig ist das integrierte TTL-Bereinigungsprogramm nicht aktiv. Sie können das TTL-Bereinigungsprogramm aktivieren, indem Sie die Methode ″setTtlEvictorType″ mit einem der folgenden drei Werte aufrufen: CREATION_TIME, LAST_ACCESS_TIME oder NONE. Der Wert CREATION_TIME zeigt an, dass das Bereinigungsprogramm TimeToLive-Attributwert der Zeit hinzufügt, zu der der Map-Eintrag in der BackingMap erstellt wurde, um zu bestimmen, wann der MapEintrag aus der BackingMap entfernt werden muss. Der Wert LAST_ACCESS_TIME zeigt an, dass das Bereinigungsprogramm den TimeToLive-Attributwert der Zeit hinzufügt, zu der der Map-Eintrag zuletzt von einer Transaktion, die in der Anwendung ausgeführt wird, aufgerufen wurde, um zu bestimmen, wann der Map-Eintrag entfernt werden muss. Der Map-Eintrag wird nur entfernt, wenn er in dem mit dem Attribut ″TimeToLive″ festgelegten Zeitraum nicht von einer Transaktion aufgerufen wird. Der Wert NONE zeigt an, dass das Bereinigungsprogramm inaktiv bleibt und keine Map-Einträge entfernt. Wenn dieses Attribut nicht gesetzt wird, wird NONE als Standardwert verwendet, und das TTL-Bereinigungsprogramms wird nicht aktiv. Einzelheiten zum integrierten TTL-Bereinigungsprogramm finden Sie in der Beschreibung der Bereinigungsprogramme.
- TimeToLive: Dieses Attribut wird verwendet, um die Anzahl der Sekunden anzugeben, die das TTL-Bereinigungsprogramm der Erstellungs- bzw. letzten Zugriffszeit für jeden Eintrag gemäß der Beschreibung des Attributs ″TtlEvictorType″ hinzufügen muss. Wenn dieses Attribut nicht gesetzt wird, wird der

Sonderwert null verwendet, um anzuzeigen, dass die Lebensdauer unbegrenzt ist. Wenn Sie dieses Attribut auf ″infinity″ setzen, werden die Map-Einträge vom Bereinigungsprogramm nicht entfernt.

Das folgende Beispiel veranschaulicht, wie Sie die BackingMap ″someMap″ in der ObjectGrid-Instanz ″someGrid″ definieren und verschiedene Attribute der BackingMap mit den set-Methoden der Schnittstelle ″BackingMap″ setzen:

```
import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.LockStrategy;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;

...

ObjectGrid og =
ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("someGrid");
BackingMap bm = objectGrid.getMap("someMap");
bm.setReadOnly( true ); // Standardeinstellung Lese-/Schreibzugriff überschreiben
bm.setNullValuesSupported(false); // Standardmäßiges Zulassen von Nullwerten überschreiben
bm.setLockStrategy( LockStrategy.PESSIMISTIC ); // Standardeinstellung OPTIMISTIC überschreiben
bm.setLockTimeout( 60 ); // Standardeinstellung von 15 Sekunden überschreiben
bm.setNumberOfBuckets(251); // Standardeinstellung überschreiben (Primzahlen funktionieren am besten)
bm.setNumberOfLockBuckets(251); // Standardeinstellung überschreiben (Primzahlen funktionieren am besten)
```

BackingMap-Plug-ins

Die Schnittstelle ″BackingMap″ hat mehrere optionale Plug-in-Punkte für erweiterbare Interaktionen mit dem ObjectGrid:

- **ObjectTransformer-Plug-in**: Für einige Map-Operationen muss eine BackingMap möglicherweise einen Schlüssel oder Wert eines Eintrags serialisieren, entserialisieren oder kopieren. Die BackingMap kann diese Aktion über eine Standardimplementierung der Schnittstelle ″ObjectTransformer″ ausführen. Eine Anwendung kann die Leistung verbessern, indem sie ein angepasstes ObjectTransformer-Plug-in bereitstellt, das von der BackingMap zum Serialisieren, Entserialisieren oder Kopieren des Schlüssels oder Werts eines Eintrags verwendet. Weitere Informationen finden Sie in den Informationen zum ObjectTransformer-Plug-in im *Programmierhandbuch*.
- **Evictor-Plug-in**: Das integrierte TTL-Bereinigungsprogramm verwendet einen zeitbasierten Algorithmus, um zu entscheiden, wann eine Eintrag aus der BackingMap entfernt werden muss. Einige Anwendungen müssen möglicherweise einen anderen Algorithmus für das Entfernen eines Eintrags aus einer BackingMap verwenden. Das Evictor-Plug-in stellt der BackingMap ein angepasstes Bereinigungsprogramm zur Verfügung. Das Evictor-Plug-in ist eine Erweiterung des integrierten TTL-Bereinigungsprogramms. Es ist kein Ersatz für das TTL-Bereinigungsprogramm. ObjectGrid stellt ein angepasstes Evictor-Plug-in bereit, das bekannte Algorithmen wie LRU (Least Recently Used) oder LFU (Least Frequently Used) implementiert. Anwendungen können eines der bereitgestellten Evictor-Plug-ins integrieren oder ein eigenes Evictor-Plug-in bereitstellen. Weitere Informationen finden Sie in den Informationen zur Bereinigung im *Programmierhandbuch*.
- **MapEventListener-Plug-in**: Eine Anwendung möchte möglicherweise über BackingMap-Ereignisse, wie z. B. das Entfernen von Map-Einträgen oder den Abschluss des Preload-Prozesses für die BackingMap, benachrichtigt werden. Eine BackingMap ruft Methoden im MapEventListener-Plug-in auf, um eine Anwendung über BackingMap-Ereignisse zu benachrichtigen. Eine Anwendung kann Benachrichtigungen über verschiedene BackingMap-Ereignisse empfangen, indem sie die Methode ″setMapEventListener″ verwendet, um der BackingMap ein oder mehrere angepasste MapEventListener-Plug-ins bereitzustellen. Die Anwendung kann die aufgelisteten MapEventListener-Objekte mit der Methode ″addMapEventListener″ oder mit der Methode ″removeMapEventListener″

ändern. Weitere Informationen finden Sie in den Informationen zum MapEvent-Listener-Plug-in im *Programmierhandbuch*.

- **Loader-Plug-in**: Eine BackingMap ist ein Speichercache einer Map. Ein Loader-Plug-in ist eine Option, die von der BackingMap verwendet wird, um Daten im Hauptspeicher zu verschieben, und wird als persistenter Speicher für die BackingMap verwendet. Es kann beispielsweise ein JDBC-Loader (Java Database Connectivity) verwendet werden, um Daten in eine BackingMap und eine oder mehrere relationale Tabellen einer relationalen Datenbank bzw. aus diesen zu verschieben. Es muss keine relationale Datenbank als persistenter Speicher für eine BackingMap verwendet werden. Der Loader kann auch verwendet werden, um Daten zwischen einer BackingMap und einer Datei, zwischen einer BackingMap und einer Hibernate-Map, zwischen einer BackingMap und einer JEE-Entity-Bean (Java 2 Platform, Enterprise Edition), zwischen einer BackingMap und einem anderen Anwendungsserver usw. zu verschieben. Die Anwendung muss ein angepasstes Loader-Plug-in bereitstellen, um Daten zwischen der BackingMap und dem persistenten Speicher für jede verwendete Technologie zu verschieben. Wenn kein Loader bereitgestellt wird, ist die BackingMap ein einfacher Speichercache. Weitere Informationen finden Sie in der Beschreibung der Verwendung von Loadern im *Programmierhandbuch*.
- **OptimisticCallback-Plug-in**: Wenn das Attribut "LockStrategy" für eine BackingMap auf OPTIMISTIC gesetzt ist, muss die BackingMap oder ein Loader-Plug-in Vergleichsoperationen für die Werte der Map durchführen. Das OptimisticCallback-Plug-in wird von der BackingMap bzw. dem Loader für die Durchführung von Vergleichsoperationen für eine optimistische Versionssteuerung verwendet. Weitere Informationen finden Sie in der Beschreibung des OptimisticCallback-Plug-ins im *Programmierhandbuch*.
- **MapIndexPlugin-Plug-in**: Ein MapIndexPlugin-Plug-in (oder kurz Index) ist eine Option, die die BackingMap verwendet, um einen Index zu erstellen, der auf dem angegebenen Attribut des gespeicherten Objekts basiert. Der Index ermöglicht der Anwendung, Objekte anhand eines bestimmten Werts oder Wertebereichs zu finden. Es gibt zwei Typen von Indizes: statische und dynamische. Ausführliche Informationen finden Sie in der Beschreibung der Indexierung.

Weitere Informationen zu Plug-ins finden Sie in der Einführung zu Plug-ins im *Programmierhandbuch*.

Sperren von Map-Einträgen

Eine ObjectGrid-BackingMap unterstützt mehrere Sperrstrategien für die Verwaltung der Konsistenz von Cacheeinträgen.

Jede BackingMap kann für die Verwendung einer der folgenden Sperrstrategien konfiguriert werden:

1. Optimistischer Sperrmodus
2. Pessimistischer Sperrmodus
3. Ohne

Die Standardsperrstrategie ist OPTIMISTIC (optimistisch). Verwenden Sie optimistisches Sperren, wenn die Daten nur selten geändert werden. Sperren werden nur für kurze Dauer gehalten, während die Daten aus dem Cache gelesen und in die Transaktion kopiert werden. Wenn der Transaktionscache mit dem Hauptcache synchronisiert wird, werden alle zwischengespeicherten Objekte, die aktualisiert wurden, mit der Originalversion verglichen. Wenn die Prüfung negativ ausfällt, wird eine Rollback-Operation für die Transaktion durchgeführt, und es wird eine Ausnahme des Typs "OptimisticCollisionException" ausgegeben.

Bei der Sperrstrategie PESSIMISTIC (pessimistisch) werden Sperren für Cacheeinträge angefordert. Diese Strategie sollte verwendet werden, wenn die Daten häufig geändert werden. Jedesmal, wenn ein Cacheeintrag gelesen wird, wird eine Sperre angefordert und so lange gehalten, bis die Transaktion abgeschlossen wird. Die Dauer einiger Sperren kann über die verfügbaren Isolationsstufen für die Sitzung optimiert werden.

Wenn keine Sperren erforderlich sind, weil die Daten nie oder nur in ruhigen Phasen aktualisiert werden, können Sie die Sperren inaktivieren, indem Sie die Sperrstrategie NONE (Ohne) konfigurieren. Diese Strategie ist sehr schnell, weil kein Sperrenmanager erforderlich ist. Die Sperrstrategie NONE eignet sich ideal für Suchtabellen und schreibgeschützte Maps.

Weitere Informationen zu Sperrstrategien finden Sie in der Dokumentation zu Sperrstrategien im Handbuch *Produktübersicht*.

Sperrstrategie festlegen

Das folgende Beispiel demonstriert, wie für die BackingMaps ″map1″, ″map2″ und ″map3″ jeweils eine andere Sperrstrategie festgelegt wird. Das erste Snippet veranschaulicht die Verwendung von XML für die Konfiguration der Sperrstrategie und das zweite Snippet einen programmgesteuerten Ansatz.

XML-Ansatz

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

    <objectGrids>
        <objectGrid name="test">
            <backingMap name="map1"
                lockStrategy="PESSIMISTIC" numberOfLockBuckets="31"/>
            <backingMap name="map2"
                lockStrategy="OPTIMISTIC" numberOfLockBuckets="409"/>
            <backingMap name="map3"
                lockStrategy="NONE"/>
        </objectGrid>
    </objectGrids>
</objectGridConfig>
```

Programmgesteuerter Ansatz

```
BackingMap-Konfiguration - Programmgesteuertes Beispiel
import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.LockStrategy;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
...
ObjectGrid og =
 ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test");
BackingMap bm = og.defineMap("map1");
bm.setLockStrategy( LockStrategy.PESSIMISTIC );
bm.setNumberOfLockBuckets(31);
bm = og.defineMap("map2");
bm.setNumberOfLockBuckets(409);
bm.setLockStrategy( LockStrategy.OPTIMISTIC );
bm = og.defineMap("map3");
bm.setLockStrategy( LockStrategy.NONE );
```

Um zu verhindern, dass eine Ausnahme des Typs ″java.lang.IllegalStateException″ ausgelöst wird, muss die Methode ″ setLockStrategy″ vor dem Aufruf der Methode ″initialize″ bzw. ″getSession″ in der lokalen ObjectGrid-Instanz aufgerufen werden.

Konfiguration des Sperrenmanagers

Wenn Sie die Sperrstrategie PESSIMISTIC oder OPTIMISTIC verwenden, wird ein Sperrenmanager für die BackingMap erstellt. Der Sperrenmanager verwendet eine

Hash-Tabelle, um die Einträge zu verfolgen, die von einer oder mehreren Transaktionen gesperrt werden. Wenn die Hash-Tabelle viele Map-Einträge enthält, kann durch die Verwendung weiterer Sperr-Buckets eine bessere Leistung erzielt werden. Das Risiko von Java-Synchronisationskollisionen sinkt mit zunehmender Anzahl an Buckets. Werden zusätzliche Buckets verwendet, sind auch mehr gemeinsame Zugriffe möglich. Die vorherigen Beispiele haben veranschaulicht, wie eine Anwendung die Anzahl der Sperr-Buckets für eine bestimmte BackingMap-Instanz festlegen kann.

Um zu verhindern, dass eine Ausnahme des Typs ″java.lang.IllegalStateException″ ausgelöst wird, muss die Methode ″ setNumberOfLockBuckets″ vor dem Aufruf der Methode ″initialize″ bzw. ″getSession″ in der ObjectGrid-Instanz aufgerufen werden. Der Methodenparameter ″setNumberOfLockBuckets″ ist ein primitiver Java-Integer, der die Anzahl der zu verwendenden Sperr-Buckets angibt. Die Verwendung einer Primzahl gewährleistet eine gleichmäßige Verteilung der Map-Einträge auf die Sperr-Buckets. Ein guter Ausgangspunkt für das Erzielen der besten Leistung ist, die Anzahl der Sperr-Buckets auf ungefähr zehn Prozent der erwarteten Anzahl an BackingMap-Einträgen zu setzen.

Sperrstrategie konfigurieren

Sie können eine optimistische Strategie, eine pessimistische Strategie oder eine Strategie ohne Sperren für jede BackingMap in der WebSphere eXtreme Scale-Konfiguration definieren.

Warum und wann dieser Vorgang ausgeführt wird

Sie können eine Sperrstrategie programmgesteuert oder mit XML konfigurieren. Weitere Informationen zu Sperren finden Sie in den Beschreibungen der Sperrstrategien im Handbuch *Produktübersicht*.

- **Optimistische Sperrstrategie konfigurieren**
  - Über das Programm

Optimistische Sperrstrategie über das Programm angeben

```
import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.LockStrategy;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
...
ObjectGrid og =
 ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test");
BackingMap bm = og.defineMap("optimisticMap");
bm.setLockStrategy( LockStrategy.OPTIMISTIC );
```

  - Mit XML

Optimistische Sperrstrategie mit XML angeben

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

- **Pessimistische Sperrstrategie konfigurieren**
  - Über das Programm

Pessimistische Sperrstrategie über das Programm angeben

```
import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.LockStrategy;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
...
ObjectGrid og =
 ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test");
BackingMap bm = og.defineMap("pessimisticMap");
bm.setLockStrategy( LockStrategy.PESSIMISTIC );
```

– Mit XML

Pessimistische Sperrstrategie mit XML angeben

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

    <objectGrids>
        <objectGrid name="test">
            <backingMap name="pessimisticMap"
                lockStrategy="PESSIMISTIC"/>
        </objectGrid>
    </objectGrids>
</objectGridConfig>
```

- **Strategie ohne Sperren konfigurieren**
  – Über das Programm

Strategie ohne Sperren über das Programm angeben

```
import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.LockStrategy;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
...
ObjectGrid og =
 ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("test");
BackingMap bm = og.defineMap("noLockingMap");
bm.setLockStrategy( LockStrategy.NONE );
```

– Mit XML

Strategie ohne Sperren mit XML angeben

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

    <objectGrids>
        <objectGrid name="test">
            <backingMap name="noLockingMap"
                lockStrategy="NONE"/>
        </objectGrid>
    </objectGrids>
</objectGridConfig>
```

Nächste Maßnahme

Um zu vermeiden, dass eine Ausnahme des Typs ″java.lang.IllegalStateException″ ausgelöst wird, müssen Sie die Methode ″setLockStrategy″ vor den Methoden ″initialize″ und ″getSession″ für die ObjectGrid-Instanz aufrufen.

Verteiltes eXtreme-Scale-Grid konfigurieren

Verwenden Sie den XML-Implementierungsrichtliniendeskriptordatei, um Ihre Implementierungstopologie zu verwalten.

Die Implementierungsrichtlinie wird als XML-Datei codiert, die dem eXtreme-Scale-Container bereitgestellt wird. Die XML-Datei enthält die folgenden Informationen:

- die Maps, die zu den einzelnen MapSets gehören,
- die Anzahl der Partitionen,
- die Anzahl synchroner und asynchroner Replikate.

Informationen zum Starten von Containerservern finden Sie im Abschnitt „Containerprozesse in einer Umgebung mit WebSphere Application Server starten″ auf Seite 244 oder im Abschnitt „Containerprozesse starten″ auf Seite 232.

Die Implementierungsrichtlinie steuert auch die folgenden Verteilungsverhalten:

- die Mindestanzahl aktiver Container, bevor die Verteilung stattfindet,
- die automatische Verteilung verloren gegangener Shards,
- die Verteilung jedes Shards einer einzelnen Partition an eine jeweils andere Maschine.

Weitere Informationen zur XML-Implementierungsrichtliniendatei finden Sie im Abschnitt „XML-Deskriptordatei für Implementierungsrichtlinie" auf Seite 151.

Endpunktinformationen werden in der dynamischen Umgebung nicht vorkonfiguriert. Es sind keine Servernamen oder Informationen zur physischen Topologie in der Implementierungsrichtlinie enthalten. Alle Shards in einem Grid werden automatisch vom Katalogservice an die Container verteilt. Der Katalogservice verwendet die in der Implementierungsrichtlinie definierten Vorgaben, um die Verteilung der Shards automatisch zu verwalten. Mit dieser automatischen Shard-Verteilung lassen sich große Grids ohne großen Aufwand konfigurieren. Sie können der Umgebung bei Bedarf auch Server hinzufügen.

Anmerkung: In einer Umgebung mit WebSphere Application Server werden Stammgruppen mit mehr als 50 Membern nicht unterstützt.

Eine XML-Implementierungsrichtliniendatei wird während des Starts an einen eXtreme-Scale-Server übergeben. Eine Implementierungsrichtlinie muss zusammen mit einer ObjectGrid-XML-Datei verwendet werden. Die Implementierungsrichtlinie ist zum Starten eines Containers zwar nicht erforderlich, aber empfehlenswert. Die Implementierungsrichtlinie muss mit der verwendeten ObjectGrid-XML-Datei kompatibel sein. Für jedes objectgridDeployment-Element in der Implementierungsrichtlinie muss ein entsprechendes objectGrid-Element in der ObjectGrid-XML vorhanden sein. Die Maps im objectgridDeployment-Element müssen mit den backingMap-Elementen in der ObjectGrid-XML konsistent sein. Jedes backingMap-Element darf nur in einem einzigen mapSet-Element referenziert werden.

Im folgenden Beispiel soll die Datei `companyGridDpReplication.xml` mit der entsprechenden Datei `companyGrid.xml` gepaart werden:

companyGridDpReplication.xml

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

 <objectgridDeployment objectgridName="CompanyGrid">
  <mapSet name="mapSet1" numberOfPartitions="11"
   minSyncReplicas="1" maxSyncReplicas="1"
   maxAsyncReplicas="0" numInitialContainers="4">
   <map ref="Customer" />
   <map ref="Item" />
   <map ref="OrderLine" />
   <map ref="Order" />
  </mapSet>
 </objectgridDeployment>

</deploymentPolicy>
```

companyGrid.xml

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

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

</objectGridConfig>
```

Die Datei `companyGridDpReplication.xml` enthält ein MapSet, das in 11 Partitionen eingeteilt ist. Jede Partition muss genau ein synchrones Replikat haben. Die Anzahl der synchronen Replikate wird über die Attribute ″minSyncReplicas″ und ″maxSyncReplicas″ vorgegeben. Da das Attribut ″minSyncReplicas″ auf 1 gesetzt ist, muss jede Partition im MapSet mindestens ein verfügbares synchrones Replikat für die Verarbeitung von Schreiboperationen haben. Da ″maxSyncReplicas″ auf 1 gesetzt ist, darf jede Partition maximal ein einziges synchrones Replikat haben. Die Partitionen in diesem MapSet haben keine asynchronen Replikate.

Das Attribut ″numInitialContainers″ weist den Katalogservice an, die Verteilung zu verzögern, bis vier Container für die Unterstützung dieses ObjectGrids verfügbar sind. Das Attribut ″numInitialContainers″ wird ignoriert, sobald die angegebene Anzahl an Containern erreicht ist.

Die Datei `companyGridDpReplication.xml` demonstriert eine gängige Methode für die Konfiguration einer Implementierungsrichtlinie, aber eine Implementierungsrichtlinie bietet Ihnen noch mehr Steuerungselemente, mit denen Sie steuern können, wie und wann Ihr ObjectGrid in der Umgebung implementiert wird. Eine vollständige Beschreibung der Implementierungsrichtliniendeskriptordatei finden Sie im Abschnitt „XML-Deskriptordatei für Implementierungsrichtlinie” auf Seite 151.

Verteilte Topologie

Verteilte kohärente Caches bieten eine höhere Leistung, Verfügbarkeit und Skalierbarkeit, die Sie konfigurieren können.

WebSphere eXtreme Scale führt eine automatische gleichmäßige Verteilung der Server durch. Sie können weitere Server hinzufügen, ohne WebSphere eXtreme Scale erneut starten zu müssen. Das Hinzufügen zusätzlicher Server ohne Neustart von eXtreme Scale ermöglicht Ihnen die Verwendung einfacher Implementierungen und Implementierungen in Terabytegröße, in denen Tausende von Servern benötigt werden. Diese Implementierungstopologie ist flexibel. Mit dem Katalogservice können Sie Server hinzufügen und entfernen, um Ressourcen besser zu nutzen, ohne den gesamten Cache entfernen zu müssen. Zum Hinzufügen oder Entfernen eines Servers verwenden Sie den Befehl startOgServer mit der Option **-catalogServiceEndPoints**, um einen Containerserver zu starten, der mit dem Katalogservice kommuniziert. Alle Clients der verteilten Topologie kommunizieren mit dem Katalogservice über Internet Interoperability Object Protocol (IIOP). Alle Clients verwenden die Schnittstelle ″ObjectGrid″, um mit Servern zu kommunizieren.

Mit den dynamischen Konfigurationsfunktionen von WebSphere eXtreme Scale können dem System Ressourcen ohne großen Aufwand hinzugefügt werden. Container enthalten die Daten, und der Katalogservice funktioniert als Touchpoint für das Grid. Die Container sind für die Verwaltung der Daten zuständig. Der Katalogservice ist für die Weiterleitung der Anforderungen an die richtige Stelle bei der ersten Berührung, die Zuordnung von Speicherplatz in Hostcontainern und die Verwaltung von Vitalität (Status) und Verfügbarkeit des Gesamtsystems zuständig. Clients stellen eine Verbindung zu einem Katalogservice her, rufen eine Beschreibung der Containerservertopologie ab und kommunizieren dann bei Bedarf direkt mit jedem Server. Wenn sich die Servertopologie durch das Hinzufügen neuer Server oder durch den Ausfall anderer Server ändert, wird der Client automatisch an den entsprechenden Server weitergeleitet, der die Daten enthält.

Ein Katalogservice existiert gewöhnlich in einem eigenen Grid von Java Virtual Machines. Für die Verwaltung mehrerer Server kann ein einziger Katalogservice

verwendet werden. Ein Container kann eigenständig in einer JVM gestartet oder zusammen mit anderen Containern für verschiedene Server in eine beliebige JVM geladen werden. Ein Client kann in jeder JVM ausgeführt werden und mit einem oder mehreren Servern kommunizieren. Ein Client kann auch in derselben JVM wie ein Container ausgeführt werden.

Sie können eine Implementierungsrichtlinie auch über das Programm erstellen, wenn Sie einen Container in einen vorhandenen Java-Prozess oder in eine vorhandene Anwendung integrieren. Weitere Informationen finden Sie in der Beschreibung der eXtreme-Scale-API "DeploymentPolicy".

eXtreme-Scale-Client konfigurieren

Sie können einen eXtreme-Scale-Client auf der Basis Ihrer Anforderungen, einschließlich Korrektureinstellungen, konfigurieren.

Sie können einen eXtreme-Scale-Client auf die folgenden Arten konfigurieren:

- XML-Konfiguration
- Programmgesteuerte Konfiguration
- Konfiguration des Spring-Frameworks
- Nahen Cache inaktivieren

Sie können die folgenden Plug-ins in einem Client überschreiben:

- **ObjectGrid-Plug-ins**
  - TransactionCallback-Plug-in
  - ObjectGridEventListener-Plug-in
- **BackingMap-Plug-ins**
  - Evictor-Plug-in
  - MapEventListener-Plug-in
  - Attribut "numberOfBuckets"
  - Attribut "ttlEvictorType"
  - Attribut "timeToLive"

Konfiguration des Clients mit XML

Zum Ändern von Einstellungen auf der Clientseite kann eine ObjectGrid-XML-Datei verwendet werden. Wenn Sie die Einstellungen in einem eXtreme-Scale-Client müssen Sie eine ObjectGrid-XML-Datei erstellen, die in ihrer Struktur der Datei gleicht, die für den eXtreme-Scale-Server verwendet wurde.

Angenommen, die folgende XML-Datei wurde mit einer XML-Implementierungsrichtliniendatei kombiniert und zum Starten eines eXtreme-Scale-Servers verwendet.

companyGridServerSide.xml

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

    <objectGrids>
        <objectGrid name="CompanyGrid">
            <bean id="TransactionCallback"
                classname="com.company.MyTxCallback" />
            <bean id="ObjectGridEventListener"
                classname="com.company.MyOgEventListener" />
            <backingMap name="Customer"
                pluginCollectionRef="customerPlugins" />
```

```
            <backingMap name="Item" />
            <backingMap name="OrderLine" numberOfBuckets="1049"
                timeToLive="1600" ttlEvictorType="LAST_ACCESS_TIME" />
            <backingMap name="Order" lockStrategy="PESSIMISTIC"
                pluginCollectionRef="orderPlugins" />
        </objectGrid>
    </objectGrids>

    <backingMapPluginCollections>
        <backingMapPluginCollection id="customerPlugins">
            <bean id="Evictor"
                className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor" />
            <bean id="MapEventListener"
                className="com.company.MyMapEventListener" />
        </backingMapPluginCollection>
        <backingMapPluginCollection id="orderPlugins">
            <bean id="MapIndexPlugin"
                className="com.company.MyMapIndexPlugin" />
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

In einem eXtreme-Scale-Server verhält sich das CompanyGrid gemäß der Definition in der Datei `companyGridServerSide.xml`. Standardmäßig hat der CompanyGrid-Client dieselben Einstellungen wie das im Server ausgeführte CompanyGrid. Einige dieser Einstellungen können jedoch im Client überschrieben werden.

Erstellen Sie ein clientspezifisches ObjectGrid, um einige dieser Einstellungen zu überschreiben. Kopieren Sie die ObjectGrid-XML-Datei, die zum Öffnen des Servers verwendet wurde, und passen Sie die Kopie für die Clientseite an. Zum Entfernen eines Plug-ins aus dem Client verwenden Sie eine leere Zeichenfolge als Wert für das Attribut "className". Wenn Sie ein vorhandenes Plug-in ändern möchten, geben Sie einen neuen Wert für das Attribut "className" an. Sie können der Datei auch ein Plug-in hinzufügen, falls es sich um eines der Plug-ins handelt, die überschrieben werden können.

Wenn Sie eines der Attribute im Client setzen möchten, geben Sie einen neuen Wert an.

Die folgende ObjectGrid-XML-Datei kann verwendet werden, um einige Attribute und Plug-ins im CompanyGrid-Client anzugeben:

companyGridClientSide.xml

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

    <objectGrids>
        <objectGrid name="CompanyGrid">
            <bean id="TransactionCallback"
                className="com.company.MyClientTxCallback" />
            <bean id="ObjectGridEventListener" className="" />
            <backingMap name="Customer" numberOfBuckets="1429"
                pluginCollectionRef="customerPlugins" />
            <backingMap name="Item" />
            <backingMap name="OrderLine" numberOfBuckets="701"
                timeToLive="800" ttlEvictorType="LAST_ACCESS_TIME" />
            <backingMap name="Order" lockStrategy="PESSIMISTIC"
                pluginCollectionRef="orderPlugins" />
        </objectGrid>
    </objectGrids>

    <backingMapPluginCollections>
        <backingMapPluginCollection id="customerPlugins">
            <bean id="Evictor"
                className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor" />
            <bean id="MapEventListener" className="" />
        </backingMapPluginCollection>
        <backingMapPluginCollection id="orderPlugins">
            <bean id="MapIndexPlugin"
                className="com.company.MyMapIndexPlugin" />
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

Die Datei `companyGridClientSide.xml` überschreibt mehrere Attribute und Plug-ins im CompanyGrid-Client. Im Client ist "com.company.MyClientTxCallback" der der TransactionCallback, wohingegen der TransactionCallback, der im Server ausge-

führt wird, ″com.company.MyTxCallback″ ist. Der ObjectGridEventListener wird im Client entfernt, weil der className-Wert eine leere Zeichenfolge ist.

Für die BackingMap ″Customer″ wird der numberOfBuckets-Wert im Client auf 1429 gesetzt. Die BackingMap ″Customer″ übernimmt das Bereinigungsprogramm (Evictor) aus der Serverkonfiguration, aber der MapEventListener wird aus dem Client entfernt.

Die Attribute `numberOfBuckets` und `timeToLive` wurde auf der Clientseite der BackingMap ″OrderLine″ angepasst.

Das Attribut ″lockStrategy″ in dieser Datei wird ignoriert, egal welcher Wert angegeben wird. Das Überschreiben dieses Attribut auf Clientseite wird nicht unterstützt.

Zum Erstellen eines CompanyGrid-Clients über die Datei `companyGridClientSide.xml` übergeben Sie die ObjectGrid-XML-Datei als URL an eine der connect-Methoden im ObjectGridManager.

Client für XML erstellen

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

Client über das Programm konfigurieren

Sie können die clientseitigen ObjectGrid-Einstellungen auch über das Programm überschreiben. Erstellen Sie ein ObjectGridConfiguration-Objekt, das in der Struktur dem serverseitigen ObjectGrid gleicht. Der folgende Code erstellt ein clientseitiges ObjectGrid, das funktional äquivalent zu dem ObjectGrid ist, das mit der Datei ″companyGridClientSide.xml″ aus dem vorherigen Abschnitt erstellt wird.

Clientseitige Einstellungen über das Programm überschreiben

```
ObjectGridConfiguration companyGridConfig = ObjectGridConfigFactory
    .createObjectGridConfiguration("CompanyGrid");
Plugin txCallbackPlugin = ObjectGridConfigFactory.createPlugin(
    PluginType.TRANSACTION_CALLBACK, "com.company.MyClientTxCallback");
companyGridConfig.addPlugin(txCallbackPlugin);

Plugin ogEventListenerPlugin = ObjectGridConfigFactory.createPlugin(
    PluginType.OBJECTGRID_EVENT_LISTENER, "");
companyGridConfig.addPlugin(ogEventListenerPlugin);

BackingMapConfiguration customerMapConfig = ObjectGridConfigFactory
    .createBackingMapConfiguration("Customer");
customerMapConfig.setNumberOfBuckets(1429);
Plugin evictorPlugin = ObjectGridConfigFactory.createPlugin(PluginType.EVICTOR,
    "com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor");
customerMapConfig.addPlugin(evictorPlugin);

companyGridConfig.addBackingMapConfiguration(customerMapConfig);

BackingMapConfiguration orderLineMapConfig = ObjectGridConfigFactory
    .createBackingMapConfiguration("OrderLine");
orderLineMapConfig.setNumberOfBuckets(701);
orderLineMapConfig.setTimeToLive(800);
orderLineMapConfig.setTtlEvictorType(TTLType.LAST_ACCESS_TIME);

companyGridConfig.addBackingMapConfiguration(orderLineMapConfig);

List ogConfigs = new ArrayList();
ogConfigs.add(companyGridConfig);

Map overrideMap = new HashMap();
overrideMap.put(CatalogServerProperties.DEFAULT_DOMAIN, ogConfigs);

ogManager.setOverrideObjectGridConfigurations(overrideMap);
ClientClusterContext client = ogManager.connect(catalogServerAddresses, null, null);
ObjectGrid companyGrid = ogManager.getObjectGrid(client, objectGridName);
```

Es werden nur die ObjectGridConfiguration- und BackingMapConfiguration-Objekte, die in der BackingMap ″overrideMap″ enthalten sind, auf überschriebene Plug-ins und Attribute hin überprüft. Der vorherige Code überschreibt beispielsweise die Anzahl der Buckets in der Map ″OrderLine″. Die Map ″Order″ bleibt auf der Clientseite jedoch unverändert, weil keine entsprechende Konfiguration eingeschlossen ist.

Client in Spring Framework konfigurieren

Clientseitige ObjectGrid-Einstellungen können auch über Spring Framework überschrieben werden. Die folgende XML-Beispieldatei veranschaulicht, wie eine ObjectGridConfiguration erstellt und zum Überschreiben einige clientseitiger Einstellungen verwendet wird. Diese Beispieldatei ruft dieselben APIs auf, die auch in der Konfiguration über das Programm verwendet werden. Das Beispiel ist funktional äquivalent zu zu dem Beispiel in der ObjectGrid-XML-Konfiguration.

Clientkonfiguration mit Spring

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
  "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
  <bean id="companyGrid" factory-bean="manager" factory-method="getObjectGrid"
    singleton="true">
    <constructor-arg type="com.ibm.websphere.objectgrid.ClientClusterContext">
      <ref bean="client" />
    </constructor-arg>
    <constructor-arg type="java.lang.String" value="CompanyGrid" />
</bean>

  <bean id="manager" class="com.ibm.websphere.objectgrid.ObjectGridManagerFactory"
    factory-method="getObjectGridManager" singleton="true">
    <property name="overrideObjectGridConfigurations">
      <map>
        <entry key="DefaultDomain">
          <list>
            <ref bean="ogConfig" />
          </list>
        </entry>
      </map>
    </property>
  </bean>

  <bean id="ogConfig"
    class="com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory"
    factory-method="createObjectGridConfiguration">
    <constructor-arg type="java.lang.String">
      <value>CompanyGrid</value>
    </constructor-arg>
    <property name="plugins">
      <list>
   <bean class="com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory"
          factory-method="createPlugin">
          <constructor-arg type="com.ibm.websphere.objectgrid.config.PluginType"
            value="TRANSACTION_CALLBACK" />
          <constructor-arg type="java.lang.String"
            value="com.company.MyClientTxCallback" />
        </bean>
        <bean class="com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory"
          factory-method="createPlugin">
          <constructor-arg type="com.ibm.websphere.objectgrid.config.PluginType"
            value="OBJECTGRID_EVENT_LISTENER" />
          <constructor-arg type="java.lang.String" value="" />
        </bean>
      </list>
 </property>
      <property name="backingMapConfigurations">
        <list>
    <bean class="com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory"
           factory-method="createBackingMapConfiguration">
           <constructor-arg type="java.lang.String" value="Customer" />
             <property name="plugins">
     <bean class="com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory"
                 factory-method="createPlugin">
                 <constructor-arg type="com.ibm.websphere.objectgrid.config.PluginType"
    value="EVICTOR" />
   <constructor-arg type="java.lang.String"
                   value="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" />
               </bean>
             </property>
             <property name="numberOfBuckets" value="1429" />
           </bean>
```

```
          <bean class="com.ibm.websphere.objectgrid.config.ObjectGridConfigFactory"
            factory-method="createBackingMapConfiguration">
              <constructor-arg type="java.lang.String" value="OrderLine" />
                <property name="numberOfBuckets" value="701" />
   <property name="timeToLive" value="800" />
   <property name="ttlEvictorType">
                  <value type="com.ibm.websphere.objectgrid.
          TTLType">LAST_ACCESS_TIME</value>
   </property>
           </bean>
         </list>
       </property>
     </bean>

     <bean id="client" factory-bean="manager" factory-method="connect"
       singleton="true">
       <constructor-arg type="java.lang.String">
    <value>localhost:2809</value>
       </constructor-arg>
   <constructor-arg
         type="com.ibm.websphere.objectgrid.security.
      config.ClientSecurityConfiguration">
         <null />
       </constructor-arg>
   <constructor-arg type="java.net.URL">
     <null />
       </constructor-arg>
     </bean>
</beans>
```

Nachdem Sie die XML-Datei erstellt haben, laden Sie die Datei und erstellen das ObjectGrid mit dem folgenden Code-Snippet:

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

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

Weitere Informationen finden Sie im Abschnitt „Integration mit dem Spring-Framework" auf Seite 210.

Nahen Cache inaktivieren

Der nahe Cache wird standardmäßig aktiviert, wenn eine optimistische Sperrstrategie oder keine Sperrstrategie konfiguriert ist, und kann nicht verwendet werden, wenn eine pessimistische Sperrstrategie konfiguriert ist.

Zum Inaktivieren des nahen Caches müssen Sie das Attribut "numberOfBuckets" in der ObjectGrid-Deskriptordatei mit den Korrekturwerten des Clients auf "0" setzen.

Weitere Informationen finden Sie in der Dokumentation zum Sperren von Map-Einträgen im *Administratorhandbuch*.

Mechanismus für Clientinaktivierung aktivieren

In einer verteilten Umgebung von WebSphere eXtreme Scale gibt es auf der Clientseite standardmäßig einen nahen Cache, wenn die optimistische Sperrstrategie verwendet wird oder Sperren inaktiviert sind. Der nahe Cache enthält seine eigenen lokalen zwischengespeicherten Daten. Wenn ein Client von eXtreme Scale eine Aktualisierung festschreibt, wird diese Aktualisierung an den nahen Cache des Clients und an den Server gesendet. Andere Clients von eXtreme Scale erhalten die Aktualisierungsinformationen jedoch nicht und können daraufhin Daten haben, die nicht auf dem neuesten Stand sind.

Naher Cache

Anwendungen müssen sich dieses Problems potenziell veralteter Daten in Clients von eXtreme Scale bewusst sein. Sie können die integrierte JMS-basierte (Java Message Service) ObjectGridEventListener-Klasse "com.ibm.websphere.objectgrid.plug-

ins.builtins.JMSObjectGridEventListener″ verwenden, um den Mechanismus für Clientinaktivierung in einer verteilten eXtreme-Scale-Umgebung, einem so genannten eXtreme-Scale-Grid, zu aktivieren.

Der Mechanismus für Clientinaktivierung ist die Lösung für das Problem veralteter Daten im nahen Cache des Clients in einer verteilten eXtreme-Scale-Umgebung. Diese Mechanismus stellt sicher, dass der nahe Cache des Clients mit Servern oder anderen Clients synchronisiert wird. Aber selbst mit diesem JMS-basierten Mechanismus für Clientinaktivierung wird der nahe Cache des Clients nicht sofort aktualisiert. Es tritt eine Verzögerung auf, wenn die Laufzeitumgebung von eXtreme Scale Aktualisierungen veröffentlicht.

Es sind zwei Modelle für den Mechanismus für Clientinaktivierung in einer verteilten eXtreme-Scale-Umgebung verfügbar:

- Client/Server-Modell: In diesem Modell haben alle Serverprozesse die Rolle ″Publisher″ (Bereitsteller), die alle Transaktionsänderungen an der vorgesehenen JMS-Destination veröffentlicht. Alle Clientprozesse haben die Rolle ″Receiver″ (Empfänger) und empfangen alle Transaktionsänderungen von der vorgesehenen JMS-Destination.
- Modell mit dem Client in zwei Rollen: In diesem Modell haben alle Serverprozesse nichts mit der JMS-Destination zu tun. Alle Clientprozesse übernehmen die Rollen ″Publisher″ und ″Receiver″ für die JMS-Destinations. Transaktionsänderungen, die auf Clientseite stattfinden, werden an der JMS-Destination veröffentlicht, und alle Clients empfangen diese Transaktionsänderungen.

Weitere Informationen zum Aktivieren des Mechanismus für die Clientinaktivierung finden Sie im Abschnitt „JMS-Ereignis-Listener″ auf Seite 146.

Client/Server-Modell

In einem Client/Server-Modell haben die Server die Rolle ″JMS-Publisher″, und der Client hat die Rolle ″JMS-Receiver″.

XML-Beispiel für Client/Server-Modell

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="AgentObjectGrid">
      <bean id="ObjectGridEventListener"
        className="com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener">
        <property name="invalidationModel" type="java.lang.String" value="CLIENT_SERVER_MODEL" description="" />
        <property name="invalidationStrategy" type="java.lang.String" value="PUSH" description="" />
        <property name="mapsToPublish" type="java.lang.String" value="agent;profile;pessimisticMap" description="" />
        <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String" value="defaultTCF" description="" />
        <property name="jms_topicJndiName" type="java.lang.String" value="defaultTopic" description="" />
        <property name="jms_topicName" type="java.lang.String" value="defaultTopic" description="" />
        <property name="jms_userid" type="java.lang.String" value="" description="" />
        <property name="jms_password" type="java.lang.String" value="" description="" />
        <property name="jndi_properties" type="java.lang.String"
          value="java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory;java.naming.provider.url=
   tcp://localhost:61616;connectionFactoryNames=defaultTCF;topic.defaultTopic=defaultTopic"
          description="jndi properties" />
      </bean>

      <backingMap name="agent" readOnly="false" pluginCollectionRef="agent" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="28800" />
      <backingMap name="profile" readOnly="false" pluginCollectionRef="profile" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="pessimisticMap" readOnly="false" pluginCollectionRef="pessimisticMap" preloadMode="false"
        lockStrategy="PESSIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="excludedMap1" readOnly="false" pluginCollectionRef="excludedMap1" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="excludedMap2" readOnly="false" pluginCollectionRef="excludedMap2" preloadMode="false"
```

```
      lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
      timeToLive="2700" />
  </objectGrid>
</objectGrids>

<backingMapPluginCollections>
  <backingMapPluginCollection id="agent">
    <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.AgentObjectTransformer" />
  </backingMapPluginCollection>
  <backingMapPluginCollection id="profile">
    <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.ProfileObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      <property name="maxSize" type="int" value="2000" description="set max size for LRU evictor" />
      <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
      <property name="numberOfLRUQueues" type="int" value="50" description="set number of LRU queues" />
    </bean>
  </backingMapPluginCollection>

  <backingMapPluginCollection id="pessimisticMap" />
  <backingMapPluginCollection id="excludedMap1" />
  <backingMapPluginCollection id="excludedMap2" />
</backingMapPluginCollections>

</objectGridConfig>
```

Modell mit dem Client in zwei Rollen

In diesem Modell übernimmt jeder Client sowohl die Rolle ″JMS-Publisher″ als auch die Rolle ″JMS-Receiver″. Der Client veröffentlicht alle festgeschriebenen Transaktionsänderungen an der vorgesehenen JMS-Destination und empfängt alle festgeschriebenen Transaktionsänderungen von anderen Clients. Der Server selbst hat in diesem Modell nichts mit JMS zu tun.

XML-Beispiel mit Zweirollenmodell

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="AgentObjectGrid">
      <bean id="ObjectGridEventListener"
        className="com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener">
        <property name="invalidationModel" type="java.lang.String" value="CLIENT_AS_DUAL_ROLES_MODEL" description="" />
        <property name="invalidationStrategy" type="java.lang.String" value="PUSH" description="" />
        <property name="mapsToPublish" type="java.lang.String" value="agent;profile;pessimisticMap" description="" />
        <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String" value="defaultTCF" description="" />
        <property name="jms_topicJndiName" type="java.lang.String" value="defaultTopic" description="" />
        <property name="jms_topicName" type="java.lang.String" value="defaultTopic" description="" />
        <property name="jms_userid" type="java.lang.String" value="" description="" />
        <property name="jms_password" type="java.lang.String" value="" description="" />
        <property name="jndi_properties" type="java.lang.String"
          value="java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory;java.naming.provider.url=
     tcp://localhost:61616;connectionFactoryNames=defaultTCF;topic.defaultTopic=defaultTopic"
          description="jndi properties" />
      </bean>

      <backingMap name="agent" readOnly="false" pluginCollectionRef="agent" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="28800" />
      <backingMap name="profile" readOnly="false" pluginCollectionRef="profile" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="pessimisticMap" readOnly="false" pluginCollectionRef="pessimisticMap" preloadMode="false"
        lockStrategy="PESSIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="excludedMap1" readOnly="false" pluginCollectionRef="excludedMap1" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="excludedMap2" readOnly="false" pluginCollectionRef="excludedMap2" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
    </objectGrid>
  </objectGrids>

  <backingMapPluginCollections>
    <backingMapPluginCollection id="agent">
      <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.AgentObjectTransformer" />
    </backingMapPluginCollection>
    <backingMapPluginCollection id="profile">
      <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.ProfileObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
        <property name="maxSize" type="int" value="2000" description="set max size for LRU evictor" />
        <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
        <property name="numberOfLRUQueues" type="int" value="50" description="set number of LRU queues" />
      </bean>
    </backingMapPluginCollection>
```

```
    <backingMapPluginCollection id="pessimisticMap" />
    <backingMapPluginCollection id="excludedMap1" />
    <backingMapPluginCollection id="excludedMap2" />
  </backingMapPluginCollections>

</objectGridConfig>
```

Zeitlimit für Anforderungswiederholung konfigurieren

Bei zuverlässigen Maps können Sie ein Wiederholungszeitlimit an WebSphere eXtreme Scale übergeben. Das Zeitlimit wird zusammen mit dem Transaktionszeitlimit für die Wiederholung von Transaktionsanforderungen verwendet.

Es gibt zwei Methoden für die Konfiguration zuverlässiger Maps. Wenn ein Wert größer als null definiert wird, wird die Anforderung so lange wiederholt, bis das Zeitlimit abläuft oder ein permanenter Fehler wie eine Ausnahme des Typs ″DuplicateKeyException″ auftritt. Wird der Wert null gesetzt, wird der Modus für schnelles Fehlschlagen (fail-fast) verwendet, und eXtreme Scale versucht nicht, die Anforderung zu wiederholen.

Sie geben einen Zeitlimitwert in Millisekunden in der Clienteigenschaftendatei oder in der Sitzung an. Die Sitzungseinstellung überschreibt immer die Einstellung in der Clienteigenschaftendatei. Zur Laufzeit wird das Transaktionszeitlimit zusammen mit dem Wiederholungszeitlimit verwendet, um sicherzustellen, dass das Wiederholungszeitlimit nicht höher ist als das Transaktionszeitlimit.

Aufgrund der Variationen in der Beendigung von Transaktionen mit automatischer Festschreibung und ohne automatische Festschreibung (Transaktionen, die explizite Methoden ″begin″ und ″commit″ verwenden), sind auch die gültigen Ausnahmen für die Wiederholung verschieden.

Für Transaktionen, die innerhalb einer Sitzung aufgerufen werden, gilt die Wiederholung für CORBA-Ausnahmen des Typs ″SystemExceptions″ und eXtreme-Scale-Ausnahmen des Typs ″TargetNotAvailable″.

Für Transaktionen mit automatischer Festschreibung ist die Wiederholung für CORBA-Ausnahmen des Typs ″SystemExceptions″ und eXtreme-Scale-Ausnahmen gültig, die sich auf die Verfügbarkeit beziehen (ReplicationVotedToRollback-TransactionException, TargetNotAvailable, AvailabilityException usw.).

Weitere Informationen finden Sie im Abschnitt zur Verwendung von Sitzungen für den Zugriff auf Daten im Grid im *Programmierhandbuch*.

Anwendungs- oder andere permanente Fehler werden sofort zurückgegeben, und die Transaktion wird nicht wiederholt. Zu diesen permanenten Fehlern gehören Ausnahmen des Typs ″DuplicateKeyException″ und ″KeyNotFoundException″.

Bei der Einstellung für schnelles Fehlschlagen werden alle Ausnahmen ohne Wiederholung zurückgegeben.

Im Folgenden Sie die Ausnahmen ausführlicher beschrieben:

Ausnahmen mit Wiederholung auf Clientseite

- ReplicationVotedToRollbackTransactionException (nur bei automatischer Festschreibung)
- TargetNotAvailable
- org.omg.CORBA.SystemException

- AvailabilityException (nur bei automatischer Festschreibung)
- LockTimeoutException (nur bei automatischer Festschreibung)
- UnavailableServiceException (nur bei automatischer Festschreibung)

Weitere Ausnahmen

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

Eigenschaft ″requestRetryTimeout″ in einer Clienteigenschaftendatei festlegen

Zum Festlegen des requestRetryTimeout-Werts in einem Client müssen Sie in der „Clienteigenschaftendatei” auf Seite 203 die Eigenschaft ″requestRetryTimeout″ hinzufügen oder ändern. Die Clienteigenschaftendatei ist standardmäßig die Datei `objectGridClient.properties`. Die Eigenschaft ″requestRetryTimeout″ wird in Millisekunden festgelegt. Setzen Sie die Eigenschaft auf einen Wert größer als null, damit eine Anforderung bei Ausnahmen, für die eine Wiederholung verfügbar ist, wiederholt wird. Setzen Sie die Eigenschaft auf `0`, damit Anforderungen bei Ausnahmen ohne Wiederholung fehlschlagen. Zur Verwendung des Standardverhaltens entfernen Sie die Eigenschaft, oder setzen Sie sie auf den Wert `-1`.

objectGridClient.properties

```
# Konfiguration eines eXtreme-Scale-Clients
preferLocalProcess = false
preferLocalhost = false
requestRetryTimeout = 30000
```

Der requestRetryTimeout-Wert wird in Millisekunden angegeben. Wenn der Wert aus dem vorherigen Beispiel in einer ObjectGrid-Instanz verwendet wird, ist der requestRetryTimeout-Wert 30000 Millisekunden bzw. 30 Sekunden.

Clienteigenschaften durch programmgesteuerten Abruf einer ObjectGrid-Verbindung festlegen

Wenn Sie die Clienteigenschaften über das Programm festlegen möchten, müssen Sie zuerst eine Clienteigenschaftendatei erstellen. Im folgenden Beispiel ist die Clienteigenschaftendatei die Datei `objectGridClient.properties`. Nach dem Herstellen einer Verbindung zum ObjectGridManager legen Sie die Clienteigenschaften fest. Rufen Sie anschließend eine ObjectGrid-Instanz ab. Die Clienteigenschaften werden in dieser ObjectGrid-Instanz gesetzt. Wenn die Clienteigenschaften geändert werden, müssen Sie eine neue ObjectGrid-Instanz abrufen.

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

Beispiel für Sitzungsüberschreibung mit automatischer Festschreibung

Wenn Sie das Zeitlimit für Anforderungswiederholung in einer Sitzung festlegen oder die Clienteigenschaft ″requestRetryTimeout″ überschreiben möchten, rufen Sie die Methode ″setRequestRetryTimeout(long)″ in der Schnittstelle ″Session″ auf.

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

Diese Sitzung verwendet jetzt einen requestRetryTimeout-Wert von 30000 Millisekunden bzw. 30 Sekunden, unabhängig von dem in der Clienteigenschaftendatei festgelegten Wert. Weitere Informationen zur Schnittstelle ″Session″ finden Sie im Abschnitt Sitzungen für den Zugriff auf Daten im Grid verwenden.

Fehler in der XML-Konfiguration beheben

Manchmal ist bei der Konfiguration von eXtreme Scale ein unerwartetes Verhalten zu beobachten.

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

Abweichungen zwischen Implementierungsrichtlinie und ObjectGrid-XML-Dateien

Die Implementierungsrichtlinie und die ObjectGrid-XML-Dateien müssen übereinstimmen. Es treten Fehler auf, wenn die ObjectGrid-Namen und Map-Namen in der Implementierungsrichtlinie und in den XML-Dateien nicht identisch sind.

Ungültige BackingMap- und Map-Referenzen

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

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

Beispiel für eine ungültige Datei "ObjectGrid.xml"

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

Beispiel für eine ungültige Datei "deploymentPolicy.xml"

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

```
            <map ref="payroll"/>
            <map ref="ledger"/>
        </mapSet>
    </objectgridDeployment>
</deploymentPolicy>
```

Nachrichten

Es tritt eine Ausnahme des Typs ″ObjectGridException″ ein, die durch eine Ausnahme des Typs ″IncompatibleDeploymentPolicyException″ verursacht wird. Für das vorherige Beispiel wird die Ausnahme in Form einer Nachricht wie der folgenden angezeigt:

com.ibm.websphere.objectgrid.IncompatibleDeploymentPolicyException: The ledger map reference in the mapSet1 mapSet of accounting objectgridDeployment does not reference a valid backingMap from the ObjectGrid XML.

Wenn in der Implementierungsrichtlinie Map-Referenzen auf BackingMaps fehlen, die in der ObjectGrid-XML-Datei aufgelistet sind, wird ebenfalls eine Ausnahme des Typs ″ObjectGridException″ als Folge einer Ausnahme des Typs ″IncompatibleDeploymentPolicyException″ ausgelöst. Beispiel:

com.ibm.websphere.objectgrid.IncompatibleDeploymentPolicyException: The accounting objectGrid contains the employee backingMap in the ObjectGrid XML. This map is not referenced in a mapSet within the accounting objectGridDeployment in the deployment policy XML.

Problem

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

Lösung

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

Ungültige ObjectGrid-Namen

Der Name des ObjectGrids wird in der ObjectGrid-XML-Datei und in der XML-Implementierungsrichtliniendatei referenziert.

Nachricht

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

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

Problem

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

Lösung

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

XML-Wert eines Attributs nicht gültig

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

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

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

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

Die folgende Nachricht wird im Protokoll aufgezeichnet:

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

Fehlende Attribute oder Tags

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

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

   </objectGrids>
</objectGridConfig>
```

Die folgende Nachricht wird im Protokoll aufgezeichnet:

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

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

Syntaxfehler

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

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

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

Nicht vorhandene Plug-in-Sammlung referenzieren

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

Nachricht

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

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

Problem

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

Referenzierung eines nicht vorhandenen Attributs - XML-Beispiel

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

```
      className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor" />
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Lösung

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

XML ohne Unterstützung einer Implementierung validieren

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

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

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

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

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

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

Loader

Mit einem Loader-Plug-in von eXtreme Scale kann sich eine eXtreme-Scale-Map wie ein Speichercache für Daten verhalten, die gewöhnlich in einem persistenten Speicher auf demselben System oder einem anderen System verwaltet werden. Gewöhnlich wird eine Datenbank oder ein Dateisystem als persistenter Speicher verwendet. Es kann auch eine ferne A Java Virtual Machine (JVM) als Datenquelle verwendet werden, was die Erstellung Hub-basierter Caches mit eXtreme Scale ermöglicht. Ein Loader enthält die Logik für das Lesen aus einem und das Schreiben in einem persistenten Speicher.

Übersicht

Loader (Ladeprogramme) sind BackingMap-Plug-ins, die aufgerufen werden, wenn Änderungen an der BackingMap vorgenommen werden oder wenn die BackingMap eine Datenanforderungen nicht bedienen kann (Cachefehler). Eine Übersicht über die Interaktion von eXtreme Scale mit einem Loader finden Sie in den Informationen zu Cachingszenarios im *Produktübersicht*.

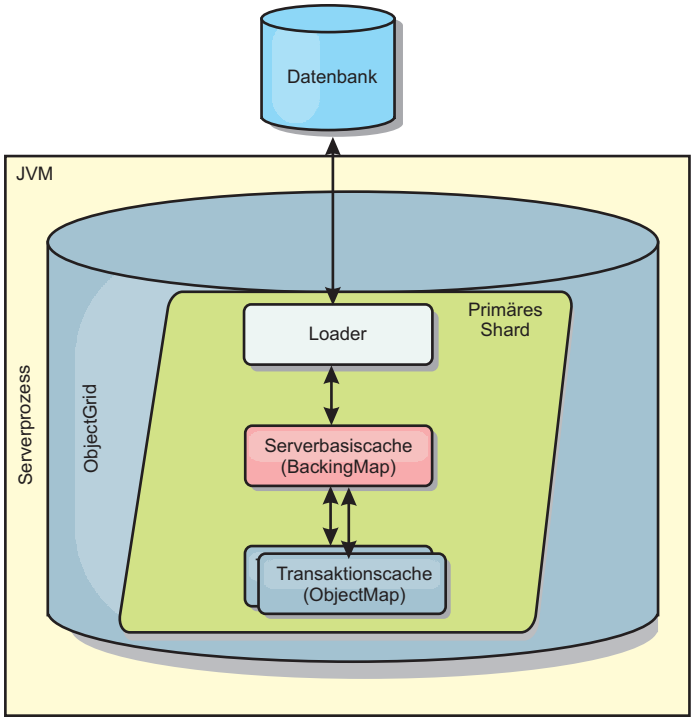


Abbildung 1. Loader

Es gibt zwei integrierte Loader, die die Integration mit relationalen Datenbank-Back-Ends erheblich vereinfachen. Die JPA-Loader nutzen die ORM-Funktionen (Object-Relational Mapping, objektbezogene Zuordnung) der OpenJPA- und Hibernate-Implementierungen der Spezifikation Java Persistence API (JPA). Weitere Informationen finden Sie in den Informationen zu JPA-Loadern im *Produktübersicht*.

Loader verwenden

Wenn Sie der BackingMap-Konfiguration einen Loader hinzufügen möchten, können Sie die programmgesteuerte Konfiguration oder die XML-Konfiguration verwenden. Ein Loader steht mit einer BackingMap in folgender Beziehung.

- Eine BackingMap kann nur einen einzigen Loader haben.
- Eine Client-BackingMap (naher Cache) kann keinen Loader haben.
- Eine Loader-Definition kann auf mehrere BackingMaps angewendet werden, aber jede BackingMap hat eine eigene Loader-Instanz.

Weitere Informationen finden Sie im Abschnitt zum Schreiben eines Loaders im Handbuch *Produktübersicht*.

XML-Konfiguration für die Integration eines Loaders

Ein anwendungsdefinierter Loader kann über eine XML-Datei integriert werden. Das folgende Beispiel veranschaulicht, wie der Loader ″MyLoader″ in die BackingMap ″map1″ integriert wird. Sie mküssen den Klassennamen für Ihren Loader, den Datenbanknamen und die Verbindungsdetails sowie die Eigenschaften für die Isolationsstufe angeben. Sie können dieselbe XML-Struktur verwenden, wenn Sie lediglich einen Preloader verwenden. Geben Sie in diesem Fall nur den Klassennamen des Preloaders an Stelle des vollständigen Loader-Klassennamens an.

Loader-Konfiguration mit XML
```
<?xml version="1.0" encoding="UTF-8" ?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
<objectGrids>
    <objectGrid name="grid">
        <backingMap name="map1" pluginCollectionRef="map1" lockStrategy="OPTIMISTIC" />
    </objectGrid>
</objectGrids>
<backingMapPluginCollections>
    <backingMapPluginCollection id="map1">
        <bean id="Loader" className="com.myapplication.MyLoader">
            <property name="dataBaseName"
                      type="java.lang.String"
                      value="testdb"
                      description="database name" />
            <property name="isolationLevel"
                      type="java.lang.String"
                      value="read committed"
                      description="iso level" />
        </bean>
    </backingMapPluginCollection>
</backingMapPluginCollections>
</objectGridConfig>
```

Loader programmgesteuert integrieren

Sie können eine programmgesteuerte Konfiguration mit lokalen speicherinternen Grids verwenden. Das folgende Code-Snippet veranschaulicht, wie ein anwendungsdefinierter Loader in die BackingMap für map1 über die API ″ObjectGrid″ integriert wird:

Programmgesteuerte Konfiguration eines Loaders
```
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.BackingMap;
ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid og = ogManager.createObjectGrid( "grid" );
BackingMap bm = og.defineMap( "map1" );
MyLoader loader = new MyLoader();
loader.setDataBaseName("testdb");
loader.setIsolationLevel("read committed");
bm.setLoader( loader );
```

In diesem Snippet wird davon ausgegangen, dass die Klasse ″MyLoader″ die anwendungsdefinierte Klasse ist, die die Schnittstelle ″com.ibm.websphere.objectgrid.plugins.Loader″ definiert. Da die Assoziation eines Loaders zu einer BackingMap nach der Initialisierung von ObjectGrid nicht mehr geändert werden kann, muss der Code vor dem Aufruf der Methode ″initialize″ der aufgerufenen ObjectGrid-Schnittstelle aufgerufen werden. Es wird eine Ausnahme vom Typ ″IllegalStateException″ in einem Aufruf der Methode ″setLoader″ ausgelöst, wenn diese nach der Initialisierung aufgerufen wird.

Der anwendungsdefinierte Loader kann definierte Eigenschaften haben. In dem Beispiel wird der Loader ″MyLoader″ verwendet, um Daten aus einer Tabelle in einer relationalen Datenbank zu lesen und Daten in diese Tabelle zu schreiben. Der

Loader muss den Namen der Datenbank und die SQL-Isolationsstufe angeben. Der Loader ″MyLoader″ enthält die Methoden ″setDataBaseName″ und ″setIsolationLevel″, mit denen die Anwendung diese beiden Loader-Eigenschaften setzen kann.

Hinweise zu Ladeprogrammen

In diesem Abschnitt werden Aspekte beschrieben die Sie beim Implementieren eines Ladeprogramms (Loader) beachten müssen.

Hinweise zum Vorherigen Laden

Loader (Ladeprogramme) sind BackingMap-Plug-ins, die aufgerufen werden, wenn Änderungen an der BackingMap vorgenommen werden oder wenn die BackingMap eine Datenanforderungen nicht bedienen kann (Cachefehler). Eine Übersicht über die Interaktion von eXtreme Scale mit einem Ladeprogramm finden Sie in den Informationen zu den integrierten Caching-Szenarios im *Produktübersicht*.

Jede BackingMap hat ein boolesches Attribut ″preloadMode″, mit dem festgelegt werden kann, ob das vorherige Laden (Preload) einer Map asynchron durchgeführt wird oder nicht. Standardmäßig ist das ″preloadMode″ auf ″false″ gesetzt, d. h., die Initialisierung der BackingMap ist erst abgeschlossen, wenn das vorherige Laden der Map abgeschlossen ist. Die Initialisierung der BackingMap ist beispielsweise erst abgeschlossen, wenn die Methode ″preloadMap″ zurückkehrt. Wenn die Methode ″preloadMap″ sehr viele Daten aus ihrem Back-End liest und in die Map lädt, kann dieser Vorgang relativ lang dauern. In diesem Fall können Sie eine BackingMap für das asynchrone vorherige Laden der Map konfigurieren, indem Sie das Attribut ″preloadMode″ auf ″true″ setzen. Diese Einstellung bewirkt, dass der Initialisierungscode der BackingMap einen Thread startet, der die Methode ″preloadMap″ aufruft. Auf diese Weise kann die Initialisierung einer BackingMap abgeschlossen werden, während das vorherige Laden der Map noch läuft.

Das folgende Code-Snippet veranschaulicht, wie das Attribut ″preloadMode″ so gesetzt wird, dass das asynchrone vorherige Laden aktiviert wird:

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

Das Attribut ″preloadMode″ kann auch über eine XML-Datei gesetzt werden, wie im folgenden Beispiel demonstriert wird:

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

TxID und die Verwendung der Schnittstelle ″TransactionCallback″

An die Methoden ″get″ und ″batchUpdate″ in der Schnittstelle ″Loader″ wird ein TxID-Objekt übergeben, das die Session-Transaktion darstellt, die die Ausführung der Operation ″get″ bzw. ″batchUpdate″ erfordert. Die Methoden ″get″ und ″batchUpdate″ können pro Transaktion mehrfach aufgerufen werden. Deshalb werden transaktionsbezogene Objekte, die der Loader benötigt, gewöhnlich in einem Slot des TxID-Objekts verwaltet. Ein JDBC-Loader (Java Database Connectivity) wird verwendet, um zu veranschaulichen, wie ein Loader die Schnittstellen ″TxID″ und ″TransactionCallback″ verwendet.

Es ist auch möglich, mehrere ObjectGrid-Maps in derselben Datenbank zu speichern. Jede Map hat einen eigenen Loader, und jeder Loader muss möglicherweise eine Verbindung zu derselben Datenbank herstellen. Wenn Verbindungen zu derselben Datenbank hergestellt werden, möchte jeder Loader dieselbe JDBC-Verbin-

dung verwenden, so dass die Änderungen an den einzelnen Tabellen im Rahmen derselben Datenbanktransaktion festgeschrieben werden. Gewöhnlich schreibt die Person, die die Loader-Implementierung schreibt, auch die TransactionCallback-Implementierung. Bei der Erweiterung der Schnittstelle ″TransactionCallback″ empfiehlt es sich, Methoden hinzuzufügen, die der Loader benötigt, um eine Datenbankverbindung anzufordern und vorbereitete Anweisungen zwischenzuspeichern. Der Grund für diese Vorgehensweise wird deutlich, wenn Sie sehen, wie die Schnittstellen ″TransactionCallback″ und ″TxID″ vom Loader verwendet werden.

Beispiel: Der Loader erfordert eine Erweiterung der Schnittstelle ″TransactionCallback″ wie die folgende:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import com.ibm.websphere.objectgrid.TxID;
public interface MyTransactionCallback extends TransactionCallback
{
    Connection getAutoCommitConnection(TxID tx, String databaseName) throws SQLException;
    Connection getConnection(TxID tx, String databaseName, int isolationLevel ) throws SQLException;
    PreparedStatement getPreparedStatement(TxID tx, Connection conn, String tableName, String sql) throws SQLException;
    Collection getPreparedStatementCollection( TxID tx, Connection conn,  String tableName );
}
```

Mit Hilfe dieser neuen Methoden können die Methoden ″get″ und ″batchUpdate″ des Loaders wie folgt eine Verbindung anfordern:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import com.ibm.websphere.objectgrid.TxID;
private Connection getConnection(TxID tx, int isolationLevel)
{
    Connection conn = ivTcb.getConnection(tx, databaseName, isolationLevel );
    return conn;
}
```

Im vorherigen Beispiel und den Beispielen, die noch folgen, sind ″ivTcb″ und ″ivOcb″ Instanzvariablen des Loaders, die gemäß der Beschreibung im Abschnitt ″Hinweise zum vorherigen Laden″ beschrieben wurden. Die Variable ″ivTcb″ ist eine Referenz auf die Schnittstelle ″MyTransactionCallback″ und die Variable ″ivOcb″eine Referenz auf die MyOptimisticCallback-Instanz. Die Variable ″databaseName″ ist eine Instanzvariable des Loaders, die als Loader-Eigenschaft während der Initialisierung der BackingMap gesetzt wurde. Das Argument ″isolationLevel″ ist eine der Konstanten der JDBC-Verbindung, die für die verschiedenen von JDBC unterstützten Isolationsstufen definiert sind. Wenn der Loader eine optimistische Implementierung nutzt, verwendet die Methode ″get″ gewöhnlich eine JDBC-Verbindung mit automatischem Festschreiben, um die Daten aus der Datenbank abzurufen. In diesem Fall kann der Loader eine Methode ″getAutoCommitConnection″ haben, die wie folgt implementiert ist:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import com.ibm.websphere.objectgrid.TxID;
private Connection getAutoCommitConnection(TxID tx)
{
    Connection conn = ivTcb.getAutoCommitConnection(tx, databaseName);
    return conn;
}
```

Rufen Sie die Methode ″batchUpdate″ wieder auf, die die folgende switch-Anweisung enthält:

```
switch ( logElement.getType().getCode() )
{
    case LogElement.CODE_INSERT:
        buildBatchSQLInsert( tx, key, value, conn );
        break;
```

```
    case LogElement.CODE_UPDATE:
        buildBatchSQLUpdate( tx, key, value, conn );
        break;
    case LogElement.CODE_DELETE:
        buildBatchSQLDelete( tx, key, conn );
        break;
}
```

Jede der buildBatchSQL-Methoden verwendet die Schnittstelle ″MyTransactionCallback″, um eine vorbereitete Anweisung abzurufen. Im Folgenden sehen Sie ein Code-Snippet, das die Methode ″buildBatchSQLUpdate″ zeigt, die eine SQL-Anweisung ″update″ für die Aktualisierung eines EmployeeRecord-Eintrags und das Hinzufügen dieses Eintrags für die Aktualisierung im Stapelbetrieb erstellt:

```
private void buildBatchSQLUpdate( TxID tx, Object key, Object value, Connection conn )
throws SQLException, LoaderException
{
    String sql = "update EMPLOYEE set LASTNAME = ?, FIRSTNAME = ?, DEPTNO = ?,
    SEQNO = ?, MGRNO = ? where EMPNO = ?";
    PreparedStatement sqlUpdate = ivTcb.getPreparedStatement( tx, conn,
   "employee", sql );
    EmployeeRecord emp = (EmployeeRecord) value;
    sqlUpdate.setString(1, emp.getLastName());
    sqlUpdate.setString(2, emp.getFirstName());
    sqlUpdate.setString(3, emp.getDepartmentName());
    sqlUpdate.setLong(4, emp.getSequenceNumber());
    sqlUpdate.setInt(5, emp.getManagerNumber());
    sqlUpdate.setInt(6, key);
    sqlUpdate.addBatch();
}
```

Nachdem die batchUpdate-Schleife alle vorbereiteten Anweisungen erstellt hat, ruft sie die Methode ″getPreparedStatementCollection″ auf. Diese Methode ist wie folgt implementiert:

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

Wenn die Anwendung die Methode ″commit″ in Session aufruft, ruft der Session-Code die Methode ″commit″ in der Methode ″TransactionCallback″ auf, nachdem sie alle von der Transaktion vorgenommenen Änderungen mit Push an den Loader für jede Map übertragen hat, die von der Transaktion geändert wurde. Da alle Loader die Methode ″MyTransactionCallback″ verwenden, um die erforderlichen Verbindungen und vorbereiteten Anweisungen abzurufen, weiß die Methode ″TransactionCallback″, welche Verbindung verwendet werden muss, um die Festschreibung der Änderungen im Back-End anzufordern. Die Erweiterung der Schnittstelle ″TransactionCallback″ mit den einzelnen von den Loadern benötigten Methoden hat somit die folgenden Vorteile:

- Das TransactionCallback-Objekt kapselt die Verwendung von TxID-Slots für transaktionsbezogene Daten, und der Loader erfordert keine Informationen zu den TxID-Slots. Der Loader muss lediglich die Methoden kennen, die TransactionCallback über die Schnittstelle ″MyTransactionCallback″ für die vom Loader benötigten unterstützenden Funktionen hinzugefügt werden.
- Das TransactionCallback-Objekt kann sicherstellen, dass die gemeinsame Verbindungsnutzung für alle Loader, die Verbindungen zu demselben Back-End herstellen, stattfindet, so dass ein zweiphasiges Festschreibungsprotokoll umgangen werden kann.
- Das TransactionCallback-Objekt kann über eine Commit- oder Rollback-Operation, die bei Bedarf in der Verbindung aufgerufen wird, sicherstellen, dass die Verbindungsherstellung zum Back-End abgeschlossen wird.
- TransactionCallback stellt sicher, dass eine Bereinigung der Datenbankressourcen stattfindet, wenn eine Transaktion abgeschlossen wird.

- TransactionCallback verheimlicht, wenn eine verwaltete Verbindung von einer verwalteten Umgebung wie WebSphere Application Server oder einem anderen J2EE-kompatiblen (Java 2 Platform, Enterprise Edition) Anwendungsserver abgerufen wird. Auf diese Weise kann derselbe Loader-Code in verwalteten und nicht verwalteten Umgebungen verwendet werden. Nur das TransactionCallback-Plug-in muss geändert werden.
- Ausführliche Informationen zur Verwendung der TxID-Slots für transaktionsbezogene Daten durch die TransactionCallback-Implementierung finden Sie in der Beschreibung des TransactionCallback-Plug-ins.

OptimisticCallback

Wie bereits erwähnt, kann der Loader einen optimistischen Ansatz für die Steuerung des gemeinsamen Zugriffs verwenden. In diesem Fall muss das Beispiel für die Methode "buildBatchSQLUpdate" geringfügig geändert werden, um einen optimistischen Ansatz zu implementieren. Es gibt verschiedene Methoden für die Verwendung eines optimistischen Ansatzes. Eine typische Methode ist die Verwendung einer Zeitmarken- oder Folgenummernspalte für die Versionssteuerung jeder Aktualisierung der Spalte. Angenommen, die Tabelle "employee" hat eine Folgenummernspalte, deren Wert jedesmal um eins erhöht wird, wenn die Zeile aktualisiert wird. In diesem Fall können Sie die Signatur der Methode "buildBatchSQLUpdate" so ändern, dass das LogElement-Objekt an Stelle des Schlüssel/Wert-Paars an sie übergeben wird. Außerdem muss sie das OptimisticCallback-Objekt verwenden, das in die BackingMap integriert wird, um das Anfangsversionsobjekt abzurufen und das Versionsobjekt zu aktualisieren. Im Folgenden sehen Sie ein Beispiel für eine geänderte Methode "buildBatchSQLUpdate", die die Instanzvariable "ivOcb" verwendet, die gemäß der Beschreibung im Abschnitt zu preloadMap initialisiert wurde:

Codebeispiel für eine geänderte Methode batch-update

```
private void buildBatchSQLUpdate( TxID tx, LogElement le, Connection conn )
 throws SQLException, LoaderException
{
    // Anfangsversionsobjekt abrufen, wenn dieser Map-Eintrag
    // in der Datenbank gelesen oder aktualisiert wurde.
Employee emp = (Employee) le.getCurrentValue();
    long initialVersion = ((Long) le.getVersionedValue()).longValue();
    // Versionsobjekt aus dem aktualisierten Employee-Objekt für die
    // SQL-Operation "update" abrufen.
Long currentVersion = (Long)ivOcb.getVersionedObjectForValue( emp );
    long nextVersion = currentVersion.longValue();
    // Jetzt die SQL-Operation "update" erstellen, die das Versionsobjekt
    // in der WHERE-Klausel für optimistische Prüfung enthält.
String sql = "update EMPLOYEE set LASTNAME = ?, FIRSTNAME = ?,
    DEPTNO = ?,SEQNO = ?, MGRNO = ? where EMPNO = ? and SEQNO = ?";
    PreparedStatement sqlUpdate = ivTcb.getPreparedStatement( tx, conn,
   "employee", sql );
    sqlUpdate.setString(1, emp.getLastName());
    sqlUpdate.setString(2, emp.getFirstName());
    sqlUpdate.setString(3, emp.getDepartmentName());
    sqlUpdate.setLong(4, nextVersion );
    sqlUpdate.setInt(5, emp.getManagerNumber());
    sqlUpdate.setInt(6, key);
    sqlUpdate.setLong(7, initialVersion);
    sqlUpdate.addBatch();
}
```

Das Beispiel zeigt, dass das LogElement-Objekt verwendet wird, um den Anfangsversionswert abzurufen. Wenn die Transaktion zum ersten Mal auf den Map-Eintrag zugreift, wird ein LogElement-Objekt mit dem Anfangs-Employee-Objekt erstellt, das aus der Map abgerufen wurde. Außerdem wird das Anfangs-Employee-Objekt an die Methode "getVersionedObjectForValue" in der Schnittstelle "OptimisticCallback" übergeben und das Ergebnis im LogElement-Objekt gespeichert. Diese Verarbeitung findet statt, bevor eine Anwendung eine Referenz auf das Anfangs-Employee-Objekt erhält und die Chance hat, eine Methode aufzurufen, die den Status des Anfangs-Employee-Objekts zu ändern.

Das Beispiel zeigt, dass der Loader die Methode ″getVersiondObjectForValue″ verwendet, um das Versionsobjekt für das aktuelle aktualisierte Employee-Objekt abzurufen. Vor dem Aufruf der Methode ″batchUpdate″ in der Schnittstelle ″Loader″ ruft eXtreme Scale die Methode ″updateVersionedObjectForValue″ in der Schnittstelle ″OptimisticCallback″ auf, um die Generierung eines neuen Versionsobjekts für das aktualisierte Employee-Objekt anzufordern. Wenn die Methode ″batchUpdate″ zu ObjectGrid zurückkehrt, wird das LogElement-Objekt mit dem aktuellen Versionsobjekt aktualisiert und als neues Anfangsversionsobjekt festgelegt. Dieser Schritt ist erforderlich, weil die Anwendung die Methode ″flush″ für die Map an Stelle der Methode ″commit″ für die Session aufgerufen haben kann. Der Loader kann von einer einzelnen Transaktion mehrfach für denselben Schlüssel aufgerufen werden. Aus diesem Grund stellt eXtreme Scale sicher, dass das LogElement-Objekt jedesmal, wenn die Zeile in der Tabelle ″employee″ aktualisiert wird, mit dem neuen Versionsobjekt aktualisiert wird.

Jetzt hat der Loader das Anfangsversionsobjekt und das Folgeversionsobjekt und kann eine SQL-Anweisung ″SQL″ ausführen, die die Spalte SEQNO auf den Wert des Folgeversionsobjekts setzt und den Wert des Anfangsversionsobjekts in der WHERE-Klausel verwendet. Dieser Ansatz wird manchmal auch als überqualifizierte update-Anweisung bezeichnet. Die Verwendung der überqualifizierten update-Anweisung ermöglicht der relationalen Datenbank sicherzustellen, dass die Zeile in der Zeit zwischen dem Lesen der Daten aus der Datenbank und dem Aktualisieren der Datenbank durch die Transaktion nicht geändert wurde. Wenn die Zeile von einer anderen Transaktion geändert wurde, gibt Zählerbereich, der von der Aktualisierung im Stapelbetrieb zurückgegeben wird, an, dass keine Zeilen für diesen Schlüssel geändert wurden. Der Loader muss sicherstellen, dass die SQL-Operation ″update″ die Zeile geändert hat. Ist dies nicht der Fall, zeigt der Loader eine Ausnahme vom Typ ″com.ibm.websphere.objectgrid.plugins.OptimisticCollisionException″ an, um das Session-Objekt darüber zu informieren, dass die Methode ″batchUpdate″ fehlgeschlagen ist, weil mehrere gleichzeitig ausgeführte Transaktionen versuchen, dieselbe Zeile in der Datenbanktabelle zu aktualisieren. Diese Ausnahme bewirkt, dass eine Rollback-Operation für das Session-Objekt durchgeführt wird, und die Anwendung muss die vollständige Transaktion wiederholen. Die Begründung ist, dass die Wiederholung erfolgreich ist, und deshalb wird dieser Ansatz auch als optimistischer Ansatz bezeichnet. Der optimistische Ansatz bietet eine bessere Leistung, wenn die Daten nur selten geändert werden und nur selten gleichzeitig ausgeführte Transaktionen versuchen, dieselbe Zeile zu aktualisieren.

Es ist wichtig, dass der Loader mit dem Parameter ″key″ des Konstruktors ″OptimisticCollisionException″ angibt, welcher Schlüssel bzw. welche Gruppe von Schlüsseln für das Fehlschlagen der optimistischen batchUpdate-Methode verantwortlich ist. Der Parameter ″key″ kann das Schlüsselobjekt selbst oder ein Bereich von Schlüsselobjekten sein, falls mehrere Schlüssel zum Fehlschlagen der optimistischen Aktualisierung geführt haben. eXtreme Scale verwendet die Methode ″getKey″ des Konstruktors ″OptimisticCollisionException″, um festzustellen, welche Map-Einträge veraltete Daten enthalten und deshalb zur Ausnahme geführt haben. Ein Teil der Rollback-Verarbeitung besteht darin, dass alle veralteten Map-Einträge aus der Map entfernt werden. Das Entfernen veralteter Einträge ist erforderlich, damit alle nachfolgenden Transaktionen, die auf dieselben Schlüssel zugreifen, bewirken, dass die Methode ″get″ der Schnittstelle ″Loader″ aufgerufen wird, um die Map-Einträge mit den aktuellen Daten aus der Datenbank zu aktualisieren.

Im Folgenden sind weitere Möglichkeiten beschrieben, mit denen ein Loader einen optimistischen Ansatz implementieren kann:

- Es ist keine Zeitmarken- oder Folgenummernspalte vorhanden. In diesem Fall gibt die Methode ″getVersionObjectForValue″ in der Schnittstelle ″OptimisticCallback″ einfach das Wertobjekt selbst als Version zurück. Bei diesem Ansatz muss der Loader eine WHERE-Klausel erstellen, die alle Felder des Anfangsversionsobjekts enthält. Dieser Ansatz ist nicht effizient, und nicht alle Spaltentypen eignen sich für die Verwendung in der WHERE-Klausel einer überqualifizierten SQL-Anweisung ″update″. Deshalb wird dieser Ansatz gewöhnlich nicht verwendet.
- Es ist keine Zeitmarken- oder Folgenummernspalte vorhanden. Anders als beim vorherigen Ansatz enthält die WHERE-Klausel jedoch die Wertfelder, die von der Transaktion geändert wurden. Eine Methode zur Erkennung der geänderten Felder ist das Einstellen des Kopiermodus in der BackingMap auf ″CopyMode-.COPY_ON_WRITE″. Dieser Kopiermodus erfordert, dass eine Value-Schnittstelle an die Methode ″setCopyMode″ in der Schnittstelle ″BackingMap″ übergeben wird. Die BackingMap erstellt dynamische Proxy-Objekte, die die angegebene Value-Schnittstelle implementieren. Mit diesem Kopiermodus kann der Loader jeden Wert in ein com.ibm.websphere.objectgrid.plugins.ValueProxyInfo-Objekt umsetzen. Die Schnittstelle ″ValueProxyInfo″ hat eine Methode, die dem Loader ermöglicht, die Liste der Attributnamen abzurufen, die von der Transaktion geändert wurden. Mit dieser Methode kann der Loader die get-Methoden in der Value-Schnittstelle für die Attributnamen aufrufen, um die geänderten Daten abzurufen und eine SQL-Anweisung ″update″ zu erstellen, die nur die geänderten Attribute setzt. Die WHERE-Klausel kann jetzt so erstellt werden, dass sie die Primärschlüsselspalte und alle geänderten Attributspalten enthält. Dieser Ansatz ist effizienter als der vorherige Ansatz, erfordert aber, dass mehr Code im Loader geschrieben werden muss, und kann einen größeren Cache für vorbereitete Anweisungen erfordern, damit die verschiedenen Permutationen behandelt werden können. Diese Einschränkung sollte jedoch kein Problem darstellen, wenn Transaktionen gewöhnlich nur wenige Attribute ändern.
- Einige relationale Datenbanken haben eine API für die Unterstützung der automatischen Verwaltung von Spaltendaten, die für eine optimistische Versionssteuerung hilfreich ist. Lesen Sie in der Dokumentation zu Ihrer Datenbank nach, ob diese Möglichkeit existiert.

JPA-Loader konfigurieren

Ein JPA-Loader (Java Persistence API (JPA)) ist eine Ladeprogramm-Plug-in-Implementierung, die JPA für die Interaktion mit der Datenbank verwendet. Zum Konfigurieren eines Loaders müssen Sie die erforderlichen Parameter konfigurieren und Aktualisierungen in Ihren XML-Konfigurationsdateien vornehmen.

Vorbereitungen

- Sie müssen eine JPA-Implementierung wie Hibernate oder OpenJPA haben.
- Als Datenbank kann jedes Back-End verwendet werden, das vom ausgewählten JPA-Provider unterstützt wird.
- Sie können das JPALoader-Plug-in verwenden, wenn Sie Daten über die API ″ObjectMap″ speichern. Verwenden Sie das JPAEntityLoader-Plug-in, wenn Sie Daten über die API ″EntityManager″ speichern.

Warum und wann dieser Vorgang ausgeführt wird

Weitere Informationen zur Funktionsweise des JPA-Loaders finden Sie in den Informationen im Handbuch *Produktübersicht*.

1. Konfigurieren Sie die Parameter, die JPA erfordert, um mit einer Datenbank zu interagieren.

Die folgenden Parameter sind erforderlich. Diese Parameter werden in der Bean ″JPALoader″ oder ″JPAEntityLoader″ und der Bean ″JPATxCallback″ konfiguriert.

- **persistenceUnitName**: Gibt den Namen der Persistenzeinheit an. Dieser Parameter wird für zwei Zwecke benötigt: zum Erstellen einer JPA-EntityManagerFactory und für das Suchen der JPA-Entitätsmetadaten in der Datei `persistence.xml`. Dieses Attribut wird in der Bean ″JPATxCallback″ gesetzt.
- **JPAPropertyFactory**: Gibt die Factory zum Erstellen einer Map für Persistenzeigenschaften an, mit denen die Standardpersistenzeigenschaften überschrieben werden sollen. Dieses Attribut wird in der Bean ″JPATxCallback″ gesetzt. Um dieses Attribut zu setzen, ist eine Spring-Konfiguration erforderlich.
- **entityClassName**: Gibt den Namen der Entitätsklasse an, die erforderlich ist, um JPA-Methoden wie EntityManager.persist, EntityManager.find usw. zu verwenden. Der JPALoader erfordert diesen Parameter, aber der Parameter ist für **JPAEntityLoader** optional. Wenn der Parameter **entityClassName** für einen JPAEntityLoader nicht konfiguriert ist, wird die in der ObjectGrid-Entitäts-Map konfigurierte Entitätsklasse verwendet. Sie müssen für den EntityManager von eXtreme Scale und den JPA-Provider dieselbe Klasse verwenden. Dieses Attribut wird in der Bean ″JPALoader″ oder ″JPAEntityLoader″ gesetzt.
- **preloadPartition**: Gibt die Partition an, bei der der Preload-Prozess für die Map gestartet wird. Wenn die Preload-Partitionsnummer kleiner als null oder größer als die Gesamtanzahl der Partitionen minus 1 ist, wird der Preload-Prozess für die Map nicht gestartet. Der Standardwert ist -1 und bedeutet, dass der Preload-Prozess standardmäßig nicht gestartet wird. Dieses Attribut wird in der Bean ″JPALoader″ oder ″JPAEntityLoader″ gesetzt.

Neben den JPA-Parametern, die in eXtreme Scale konfiguriert werden müssen, werden JPA-Metadaten verwendet, um den Schlüssel von den JPA-Entitäten abzurufen. Die JPA-Metadaten können als Annotation oder in einer Datei `orm.xml` konfiguriert werden, die in der Datei `persistence.xml` angegeben wird. Sie sind nicht Teil der Konfiguration von eXtreme Scale.

2. Konfigurieren Sie XML-Dateien für die JPA-Konfiguration.

   Informationen zum Konfigurieren eines JPALoader oder JPAEntityLoader finden Sie in den Beschreibungen von Loader-Plug-ins im *Programmierhandbuch*.

   Konfigurieren Sie ein JPATxCallback-Transaktions-Callback zusammen mit der Loader-Konfiguration. Das folgende Beispiel ist eine ObjectGrid-XML-Deskriptordatei (`objectgrid.xml`), in der ein JPAEntityLoader und ein JPATxCallback konfiguriert sind:

```
Loader mit Callback konfigurieren - XML-Beispiel
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
      <objectGrid name="JPAEM" entityMetadataXMLFile="jpaEMD.xml">
        <bean id="TransactionCallback"
           className="com.ibm.websphere.objectgrid.jpa.JPATxCallback">
           <property
              name="persistenceUnitName"
              type="java.lang.String"
              value="employeeEMPU" />
       </bean>
       <backingMap name="Employee" pluginCollectionRef="Employee" />
    </objectGrid>
  </objectGrids>
```

```
  <backingMapPluginCollections>
    <backingMapPluginCollection id="Employee">
        <bean id="Loader"
          className="com.ibm.websphere.objectgrid.jpa.JPAEntityLoader">
       <property
              name="entityClassName"
              type="java.lang.String"
              value="com.ibm.ws.objectgrid.jpa.test.entity.Employee"/>
        </bean>
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Wenn Sie eine JPAPropertyFactory konfigurieren möchten, müssen Sie eine Spring-Konfiguration verwenden. Im Folgenden sehen Sie eine XML-Musterkonfigurationsdatei mit dem Namen `JPAEM_spring.xml`, in der eine Spring-Bean für die Konfigurationen von eXtreme Scale konfiguriert wird.

Loader mit JPA-Eigenschaften-Factory konfigurieren - XML-Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

  <objectgrid:JPAEntityLoader id="jpaLoader"
  entityClassName="com.ibm.ws.objectgrid.jpa.test.entity.Employee"/>
  <objectgrid:JPATxCallback id="jpaTxCallback" persistenceUnitName="employeeEMPU" />
</beans>
```

Die XML-Konfigurationsdatei `Objectgrid.xml` folgt. Beachten Sie, dass der ObjectGrid-Name JPAEM ist und dem ObjectGrid-Namen in der Spring-Konfigurationsdatei `JPAEM_spring.xml` entspricht.

JPAEM-Loader-Konfiguration - XML-Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="JPAEM" entityMetadataXMLFile="jpaEMD.xml">
      <bean id="TransactionCallback"
            className="{spring}jpaTxCallback"/>
        <backingMap name="Employee" pluginCollectionRef="Employee"
                    writeBehind="T4"/>
      </objectGrid>
  </objectGrids>

  <backingMapPluginCollections>
    <backingMapPluginCollection id="Employee">
       <bean id="Loader" className="{spring}jpaLoader" />
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Eine Entität kann mit den JPA-Annotationen und mit den EntityManager-Annotationen von eXtreme Scale annotiert werden. Jede Annotation hat ein funktional entsprechendes XML-Element, das verwendet werden kann. Deshalb wurde in eXtreme Scale der Spring-Namespace hinzugefügt. Sie können diese Annotationen auch über die Spring-Namespace-Unterstützung konfigurieren. Weitere Einzelheiten zur Spring-Namespace-Unterstützung finden Sie im Abschnitt „Integration mit dem Spring-Framework” auf Seite 210.

Zeitbasierte JPA-Aktualisierungskomponente konfigurieren

Sie können eine zeitbasierte Datenbankaktualisierung mit XML für eine lokale oder verteilte Konfiguration von eXtreme Scale konfigurieren. Eine lokale Konfiguration kann auch programmgesteuert konfiguriert werden.

Warum und wann dieser Vorgang ausgeführt wird

Weitere Informationen zur Funktionsweise der zeitbasierten JPA-Datenaktualisierungskomponente finden Sie in den Informationen im *Programmierhandbuch*.

Erstellen Sie eine timeBasedDBUpdate-Konfiguration.

- **Mit einer XML-Datei:**

  Das folgende Beispiel zeigt eine Datei `objectgrid.xml`, die eine timeBasedDBUpdate-Konfiguration enthält:

  Zeitbasierte JPA-Aktualisierungskomponente - XML-Beispiel

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
        <objectGrid name="changeOG"
            entityMetadataXMLFile="userEMD.xml">
            <backingMap name="user" >
                <timeBasedDBUpdate timestampField="rowChgTs"
                      persistenceUnitName="userderby"
                      entityClass="com.test.UserClass"
                      mode="INVALIDATE_ONLY"
                />
            </backingMap>
        </objectGrid>
    </objectGrids>
    <backingMapPluginCollections>
</objectGridConfig>
```

  In diesem Beispiel wird die Map ″user″ mit einer zeitbasierten Datenbankaktualisierung konfiguriert. Der Datenbankaktualisierungsmodus ist INVALIDATE_ONLY, und das Zeitmarkenfeld ist ″rowChgTs″.

  Wenn das verteilte ObjectGrid ″changeOG″ im Containerserver gestartet wird, wird automatisch ein Thread für die zeitbasierte Datenbankaktualisierung in Partition 0 gestartet.

- **Programmgesteuert:**

  Wenn Sie ein lokales ObjectGrid erstellen, können Sie auch ein TimeBasedDBUpdateConfig-Objekt erstellen und in der BackingMap-Instanz definieren:

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

  Weitere Informationen zum Definieren eines Objekts in der BackingMap-Instanz finden Sie in den Informationen zur Schnittstelle ″BackingMap″ in der API-Dokumentation.

  Alternativ können Sie das Zeitmarkenfeld in der Entitätsklasse mit der Annotation ″com.ibm.websphere.objectgrid.jpa.dbupdate.annotation.Timestamp″ annotieren. Wenn Sie den Wert in der Klasse konfigurieren, müssen Sie das Zeitmarkenfeld nicht in der XML-Konfiguration konfigurieren.

Nächste Maßnahme

Starten Sie die zeitbasierte JPA-Datenaktualisierungskomponente. Weitere Informationen finden Sie in den Informationen zum Starten der zeitbasierten JAP-Datenaktualisierungskomponente im *Programmierhandbuch*.

Cache-Plug-in für JPA

WebSphere eXtreme Scale enthält Cache-Plug-ins der Stufe 2 (L2) für die JPA-Provider OpenJPA und Hibernate.

Die Verwendung von eXtreme Scale als L2-Cacheprovider erhöht sich die Leistung beim Lesen und Abfragen von Daten und reduziert die Last der Datenbank. WebSphere eXtreme Scale bietet im Vergleich mit integrierten Cache-implementierungen verschiedene Vorteile, weil der Cache automatisch in allen Prozessen repliziert wird. Wenn ein Client einen Wert zwischenspeichert, können alle anderen Clients den zwischengespeicherten Wert, der sich lokal im Speicher befindet, verwenden.

Mit den ObjectGrid-Cache-Plug-ins für OpenJPA und Hibernate können Sie drei Topologietypen erstellen: integriert, integriert-partitioniert und fern.

Integrierte Topologie

Eine integrierte Topologie erstellt einen eXtreme-Scale-Server in dem Prozessbereich jeder Anwendung. OpenJPA und Hibernate lesen die Speicherkopie des Caches direkt und schreiben in alle anderen Kopien. Sie können die Schreibleistung durch den Einsatz asynchroner Replikation verbessern. Diese Standardtopologie liefert die beste Leistung, wenn die zwischengespeicherte Datenmenge in einen einzigen Prozess passt.

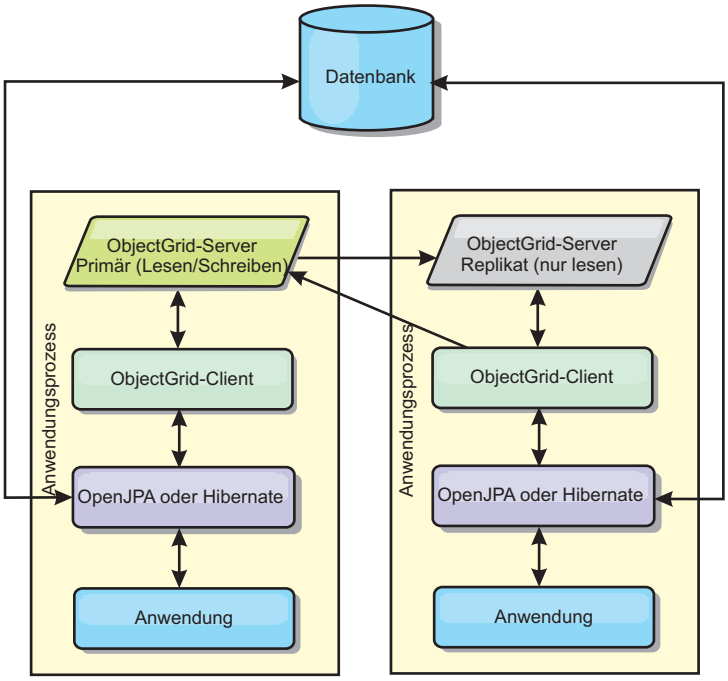


Abbildung 2. Integrierte JPA-Topologie

Vorteile:

- Alle Leseoperationen im Cache sind sehr schnelle lokale Zugriffe.
- Die Konfiguration ist einfach.

Einschränkungen:

- Das Datenvolumen ist auf die Größe des Prozesses beschränkt.
- Alle Cacheaktualisierungen werden an einen einzigen Prozess gesendet.

Integrierte, partitionierte Topologie

Wenn die zwischengespeicherten Daten nicht in einen einzigen Prozess passen, verwendet die integrierte, partitionierte Topologie ObjectGrid-Partitionen, um die Daten auf mehrere Prozesse zu verteilen. Die Leistung ist nicht so hoch wie bei der integrierten Topologie, weil die meisten Leseoperationen im Cache über Fernzugriff durchgeführt werden. Sie können diese Option trotzdem verwenden, wenn die Latenzzeit der Datenbank hoch ist.

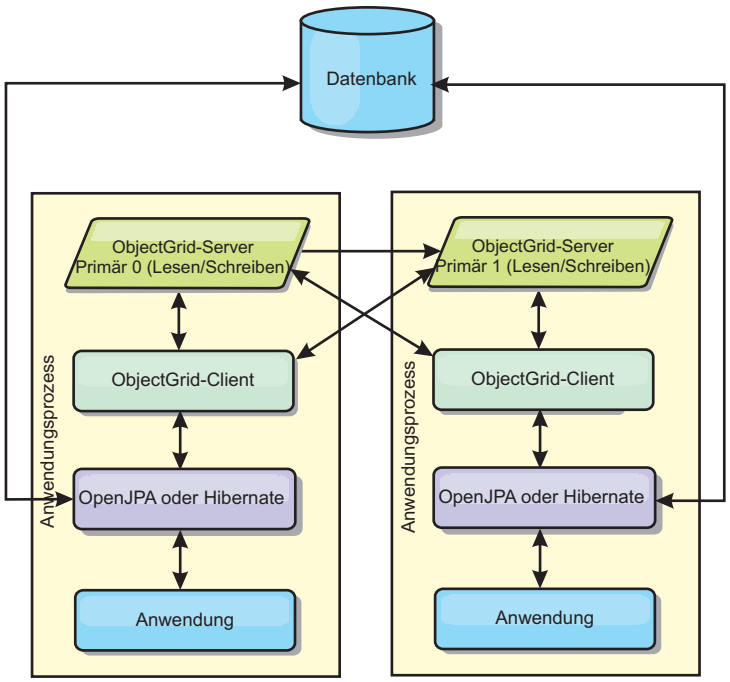


Abbildung 3. Integrierte, partitionierte JPA-Topologie

Vorteile:

- Es können große Datenvolumen gespeichert werden.
- Die Konfiguration ist einfach.
- Cacheaktualisierungen werden auf mehrere Prozesse verteilt.

Einschränkungen:

- Die meisten Lese- und Aktualisierungsoperationen im Cache werden über Fernzugriff durchgeführt.

Um beispielsweise 10 GB Daten mit maximal 1 GB pro JVM zu speichern, sind zehn Java Virtual Machines erforderlich. Die Anzahl der Partitionen muss daher auf mindestens 10 gesetzt werden. Im Idealfall wird die Anzahl der Partitionen auf eine Primzahl gesetzt, so dass in jedem Shard eine angemessene Speichermenge zugeteilt wird. Gewöhnlich entspricht der Wert der Einstellung "numberOfPartitions" der Anzahl der Java Virtual Machines. Bei dieser Einstellung enthält jede JVM eine Partition. Wenn Sie die Replikation aktivieren, müssen Sie die Anzahl der Java Virtual Machines im System erhöhen. Andernfalls wird in jeder JVM zusätzlich eine Replikatpartition gespeichert, die genauso viel Speicher belegt wie eine primäre Partition.

In einem System mit vier Java Virtual Machines und einem numberOfPartitions-Wert von 4 beispielsweise enthält jede JVM eine primäre Partition. Bei einer Lese-

operation besteht eine Chance von 25 %, dass die Daten aus einer lokal verfügbaren Partition abgerufen werden, was im Vergleich mit dem Abruf der Daten aus einer fernen JVM wesentlich schneller ist. Wenn eine Leseoperation, z. B. eine Abfrage, eine Sammlung von Daten abrufen muss, die gleichmäßig auf vier Partitionen verteilt sind, sind 75 % der Aufrufe ferne und 25 % der Aufrufe lokale Aufrufe. Wenn die Einstellung "ReplicaMode" auf `SYNC` oder `ASYNC` und die Einstellung "ReplicaReadEnabled" auf `true` gesetzt wird, werden vier Relikatpartitionen erstellt und auf vier Java Virtual Machines verteilt. Jede JVM enthält eine primäre Partition und eine Replikatpartition. Die Chance, dass die Leseoperation lokal ausgeführt wird, erhöht sich auf 50 %. Die Leseoperation, die eine Sammlung von Daten abgerufen muss, die gleichmäßig auf vier Partitionen verteilt sind, hat 50 % ferne Aufrufe und 50% lokale Aufrufe. Lokale Aufrufe sind wesentlich schneller als ferne Aufrufe. Mit jedem fernen Aufruf nimmt die Leistung ab.

Ferne Topologie

In einer fernen Topologie werden alle zwischengespeicherten Daten in einem oder mehreren gesonderten Prozessen gespeichert, was die Speicherbelegung der Anwendungsprozesse verringert. Sie können konfigurieren, dass eXtreme Scale partitioniert und repliziert wird. Eine ferne Konfiguration wird unabhängig von der Anwendung und vom JPA-Provider verwaltet.

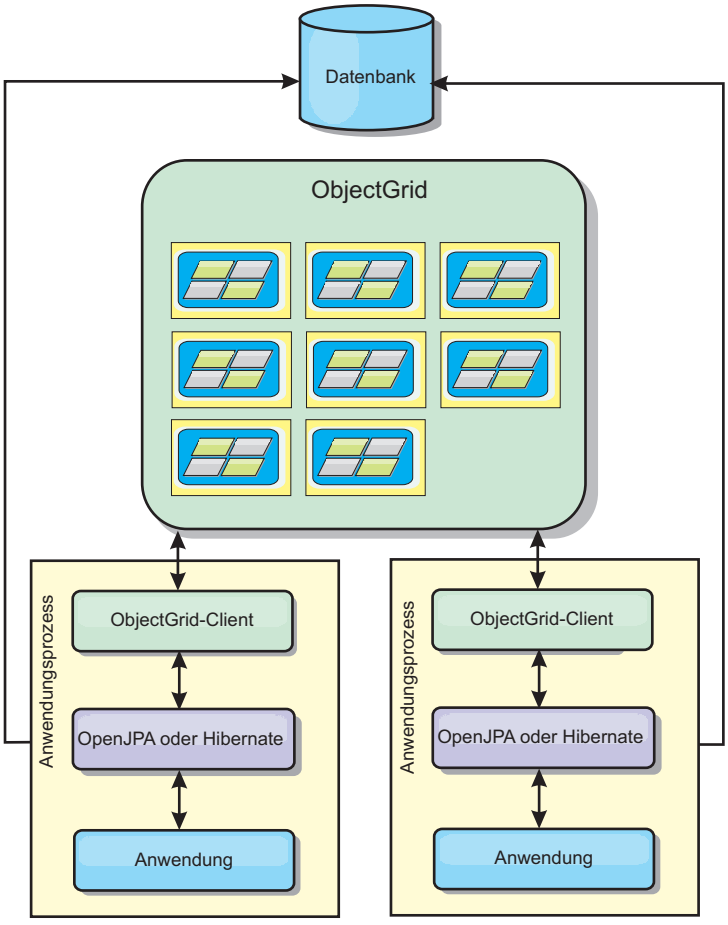


Abbildung 4. Ferne JPA-Topologie

Vorteile:

- Es können große Datenvolumen gespeichert werden.

- Der Anwendungsprozess ist frei von zwischengespeicherten Daten.
- Cacheaktualisierungen werden auf mehrere Prozesse verteilt.
- Sehr flexible Konfigurationsoptionen.

Einschränkungen:

- Alle Lese- und Aktualisierungsoperationen im Cache werden über Fernzugriff durchgeführt.

Konfiguration des JPA-Cache-Plug-ins

WebSphere eXtreme Scale enthält Cache-Plug-ins der Stufe 2 für die JPA-Provider OpenJPA und Hibernate.

Konfigurationseigenschaften des ObjectGrid-JPA-Caches

Die können das JPA-Cache-Plug-in mit den folgenden Eigenschaften konfigurieren, die alle optional sind.

ObjectGridName
Gibt den eindeutigen ObjectGrid-Namen an. Der Standardwert ist der Name der definierten Persistenzeinheit. Wenn der Name der Persistenzeinheit nicht über den JPA-Provider verfügbar ist, wird ein generierter Name verwendet.

ObjectGridType
Gibt den Typ des ObjectGrids an.

Gültige Werte:

- **EMBEDDED**: Der Standard- und empfohlene Konfigurationstyp. Zu den Standardeinstellungen gehören `NumberOfPartitions=1`, `ReplicaMode=SYNC`, `ReplicaReadEnabled=true` und `MaxNumberOfReplicas=47`. Verwenden Sie den Parameter **ReplicaMode**, um den Replikationsmodus festzulegen, und den Parameter **MaxNumberOfReplicas**, um die maximale Anzahl an Replikaten festzulegen. Wenn ein System mehr als 47 Java Virtual Machines hat, setzen Sie **MaxNumberOfReplicas** auf die Anzahl der Java Virtual Machines.
- **EMBEDDED_PARTITION**: Der zu verwendende Typ, wenn das System hohe Datenvolumen in einem verteilten System zwischenspeichern muss. Die Standardanzahl an Partitionen ist `47` beim Replikationsmodus `NONE`. In einem kleinen System, das nur wenige Java Virtual Machines hat, setzen Sie **NumberOfPartitions** auf einen Wert kleiner-gleich der Anzahl Java Virtual Machines. Sie können Werte für **ReplicaMode**, **NumberOfPartitions** und **ReplicaReadEnabled** angeben, um das System zu optimieren.
- **REMOTE**: Der Cache versucht, über den Katalogservice eine Verbindung zu einem fernen, verteilten ObjectGrid herzustellen.

NumberOfPartitions

Gültige Werte: Größer-gleich 1Gibt die Anzahl der für den Cache zu verwendenden Partitionen an. Diese Eigenschaft gilt, wenn `EMBEDDED_PARTITION` als Wert für ObjectGridType angegeben ist. Der Standardwert ist `47`. Für den Typ `EMBEDDED` ist der Wert von **NumberOfPartitions** immer 1.

ReplicaMode

Gültige Werte: SYNC/ASYNC/NONEGibt die Methode an, die verwendet wird, um den Cache in die Replikate zu kopieren. Diese Eigenschaft gilt, wenn Sie EMBEDDED oder EMBEDDED_PARTITION als Wert für ObjectGridType festgelegt haben. Der Standardwert ist NONE für den Typ EMBEDDED_PARTITION und SYNC für den Typ EMBEDDED. Wenn Sie **ReplicaMode** auf NONE und EMBEDDED für ObjectGridType angeben, verwendet der Typ EMBEDDED weiterhin den **ReplicaMode**-Wert SYNC.

ReplicaReadEnabled

Gültige Werte: TRUE oder FALSEWenn Sie diese Eigenschaft aktivieren, lesen Clients aus Replikaten. Diese Eigenschaft gilt für den Typ EMBEDDED_PARTITION. Der Standardwert ist FALSE für den Typ EMBEDDED_PARTITION. Beim Typ EMBEDDED wird **ReplicaReadEnabled** immer auf TRUE gesetzt.

MaxUsedMemory

Gültige Werte: TRUE oder FALSEAktiviert das Entfernen von Cacheeinträgen, wenn ein Speicherengpass auftritt. Der Standardwert ist TRUE und sorgt dafür, dass Daten entfernt werden, wenn die Auslastung des JVM-Heap-Speichers den Schwellenwert von 70 % überschreitet. Sie können den Prozentsatz für den Schwellwert für die Auslastung des JVM-Heap-Speichers ändern, indem Sie die Eigenschaft "memoryThresholdPercentage" in der Datei objectGridServer.properties definieren und diese Datei in den Klassenpfad stellen. Weitere Informationen zu Bereinigungsprogrammen finden Sie in der Beschreibung der Bereinigungsprogramme (Evictor) im Handbuch *Produktübersicht*. Weitere Informationen zur Servereigenschaftendatei finden Sie in der Veröffentlichung *Administratorhandbuch*.

MaxNumberOfReplicas

Gültige Werte: Größer-gleich 1Gibt die maximale Anzahl der für den Cache zu verwendenden Replikate an. Dieser Wert gilt nur für den Typ EMBEDDED. Der muss größer-gleich der Anzahl an Java Virtual Machines in einem System sein. Der Standardwert ist 47.

Die Eigenschaften "NumberOfPartitions", "ReplicaMode", "ReplicaReadEnabled" und "MaxNumberOfReplicas" sind Faktoren für die ObjectGrid-Implementierung. Die Eigenschaften "NumberOfPartitions", "ReplicaMode" und "ReplicaReadEnabled" gelten für den Typ EMBEDDED_PARTITION. ReplicaMode und dMaxNumberOfReplicas gelten beide für den Typ EMBEDDED.

Hinweise zu den Typen EMBEDDED und EMBEDDED_PARTITION

Die integrierten ObjectGrid-Typen verwenden die zuvor beschriebenen Konfigurationseigenschaften, um eine Gruppe von ObjectGrid-Containerservern und bei Bedarf einen Katalogservice zu konfigurieren und zu implementieren. Der Lebenszyklus der Container ist an die JPA-Anwendung gebunden und wird im Klassenpfad der Anwendung zusammengefasst. Wenn eine Anwendung gestartet wird, erkennt oder startet das Plug-in automatisch einen Katalogservice, startet einen Container und stellt die Verbindung zum Katalogservice her. Das Plug-in kommuniziert anschließend mit dem ObjectGrid-Container und seinen Peers, die in anderen Anwendungsserverprozessen ausgeführt werden, über die Clientverbindung.

Jeder JPA-Entität wird über den Klassennamen der Entität eine unabhängige BackingMap zugeordnet. Jede BackingMap hat die folgenden Attribute:

- readOnly="false"

- copyKey="false"
- lockStrategy="NONE"
- copyMode="NO_COPY"

Anmerkung: Wenn Sie den ObjectGridTyp-Wert EMBEDDED oder EMBEDDED_PARTITION in einer Java-SE-Umgebung verwenden, verwenden Sie am Ende des Programms die Methode "System.exit(0)", um den integrierten eXtreme-Scale-Server zu stoppen. Andernfalls scheint es so, als würde das Programm nicht reagieren.

ObjectGridType-Standardwerte

Der ObjectGridType-Wert gibt die Topologie an, in der der ObjectGrid-Cache implementiert wird. Der Standardtyp und der Typ, der die beste Leistung bietet, ist EMBEDDED. In den folgenden Abschnitten werden die Standardeigenschaften für jeden der ObjectGridType-Werte beschrieben.

Standardwerte für die ObjectGrid-JPA-Cachetopologie EMBEDDED

Wenn Sie den ObjectGrid-Typ EMBEDDED verwenden, werden die folgenden Standardwerte verwendet, wenn Sie keine Werte in der Konfiguration angeben:

- **ObjectGridName**: Name der Persistenzeinheit
- **ObjectGridType**: EMBEDDED
- **NumberOfPartitions**: 1 (kann nicht geändert werden, wenn der ObjectGrid-Typ EMBEDDED ist)
- **ReplicaMode**: SYNC
- **ReplicaReadEnabled**: TRUE (kann nicht geändert werden, wenn der ObjectGrid-Typ EMBEDDED ist)
- **MaxUsedMemory**: TRUE
- **MaxNumberOfReplicas**: 47 (muss kleiner-gleich der Anzahl Java Virtual Machines in einem verteilten System sein)

Sie müssen einen eindeutigen ObjectGridName-Wert angeben, um Namensunverträglichkeiten zu vermeiden. Der MaxNumberOfReplicas-Wert muss größer-gleich der Gesamtanzahl Java Virtual Machines im System sein.

ObjectGrid-Cachetopologie REMOTE

Der ObjectGrid-Typ REMOTE erfordert keine Eigenschaftseinstellungen, weil das ObjectGrid und die Implementierungsrichtlinie gesondert von der JPA-Anwendung definiert werden. Das JPA-Cache-Plug-in stellt über Fernzugriff eine Verbindung zu einem vorhandenen fernen ObjectGrid her.

Da alle Interaktionen mit dem ObjectGrid über Fernzugriff erfolgen, hat diese Topologie die geringste Leistung von allen ObjectGrid-Typen.

Hinweise zum Katalogservice und Konfiguration

Wenn Sie mit einer Topologie des Typs EMBEDDED oder EMBEDDED_PARTITION arbeiten, startet das JPA-Cache-Plug-in bei Bedarf automatisch einen einzigen Katalogservice in einem der Anwendungsserverprozesse. In einer Produktionsumgebung müssen Sie ein Katalogserver-Grid erstellen. Weitere Informationen zum Definieren eines Katalogservice finden Sie .in der Beschreibung des Katalogservice mit hoher Verfügbarkeit im Handbuch *Produktübersicht*.

Wenn Sie in einem Prozess von WebSphere Application Server arbeiten, stellt das JPA-Cache-Plug-in automatisch eine Verbindung zum Katalogservice bzw. Katalogservice-Grid her, der bzw. das für die Zelle von WebSphere Application Server definiert ist. Weitere Informationen zum Definieren eines Katalogservice-Grids finden Sie in den Informationen zum Starten des Katalogservice im *Administratorhandbuch*.

Wenn Sie Ihre Server nicht in einem Prozess von WebSphere Application Server ausführen, werden die Hosts und Ports des Katalogservice-Grids über eine Eigenschaftendatei mit dem Namen `objectGridServer.properties` angegeben. Diese Datei muss im Klassenpfad der Anwendung gespeichert werden und die definierte Eigenschaft **catalogServiceEndPoints** haben. Das Katalogservice-Grid wird unabhängig von den Anwendungsprozessen gestartet und muss vor den Anwendungsprozessen gestartet werden.

Das Format der Datei `objectGridServer.properties` ist wie folgt:

```
catalogServiceEndPoints=<Hostname1>:<Port1>,<Hostname2>:<Port2>
```

Konfiguration eines Hibernate-Cache-Plug-ins

Ein eXtreme-Scale-Cache kann für Hibernate aktiviert werden, indem Eigenschaften in der Konfigurationsdatei definiert werden.

Einstellungen

Im Folgenden sehen Sie die Syntax für die Definition der Eigenschaft in der Datei `persistence.xml`:

persistence.xml

```
<property name="hibernate.cache.provider_class"
         value="com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider" />
<property name="hibernate.cache.use_query_cache" value="true"/>
<property name="objectgrid.configuration" value="<Eigenschaft>=<Wert>,..." />
<property name="objectgrid.hibernate.regionNames" value="<Regionsname>,.." />
```

Im Folgenden sehen Sie die Syntax für die Definition der Eigenschaft in der Datei `hibernate.cfg.xml`:

hibernate.cfg.xml

```
<property name="cache.provider_class">com.ibm.websphere.objectgrid.
          hibernate.cache.ObjectGridHibernateCacheProvider</property>
<property name="cache.use_query_cache">true</property>
<property name="objectgrid.configuration"><Eigenschaft>=<Wert>,...</property>
<property name="objectgrid.hibernate.regionNames"><Regionsname>,...</property>
```

Die Eigenschaft für "provider_class" ist "com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider". Zum Aktivieren des Abfragecaches setzen Sie den Wert der Eigenschaft "use_query_cache" auf `true`. Verwenden Sie die Eigenschaft "objectgrid.configuration", um die Konfigurationseigenschaften für den eXtreme-Scale-Cache festzulegen.

Sie müssen einen eindeutigen Wert für die Eigenschaft "ObjectGridName" angeben, um potenzielle Namenskonflikte zu vermeiden. Die anderen Konfigurationseigenschaften für den eXtreme-Scale-Cache sind optional.

Die Eigenschaft "objectgrid.hibernate.regionNames" ist optional und muss angegeben werden, wenn nach der Initialisierung des eXtreme-Scale-Caches regionsNames-Werte definiert werden. Angenommen, eine Entitätsklasse ist einem regionName-Wert zugeordnet, und die Entitätsklasse ist weder in der Datei `persistence.xml` noch in der Hibernate-Zuordnungsdatei enthalten. Weiter angenommen, die Entitätsklasse hat eine Annotation "Entity". In diesem Fall wird der regionName-Wert für diese Entitätsklasse beim Laden der Klassen aufgelöst, wenn der eXtreme-

Scale-Cache initialisiert wird. Ein weiteres Beispiel ist die Methode ″Query.setCacheRegion(String regionName)″, die nach der Initialisierung des eXtreme Scale-Caches ausgeführt wird. In diesen Situationen müssen Sie alle möglichen dynamisch bestimmten regionNames in die Eigenschaft ″objectgrid.hibernate.regionNames″ einfügen, damit der eXtreme-Scale-Cache BackingMaps für alle regionNames vorbereiten kann.

Im Folgenden sehen Sie Beispiele für die Dateien `persistence.xml` und `hibernate.cfg.xml`:

persistence.xml

```
<persistence-unit name="testPU2">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <class>com.ibm.websphere.objectgrid.jpa.test.Department</class>
    <properties>
      <property name="hibernate.show_sql" value="true" />
      <property name="hibernate.connection.url" value="jdbc:derby:DB_testPU2;create=true" />
      <property name="hibernate.connection.driver_class" value="org.apache.derby.jdbc.EmbeddedDriver" />

      <property name="hibernate.cache.provider_class" value="com.ibm.websphere.objectgrid.
               hibernate.cache.ObjectGridHibernateCacheProvider" />
      <property name="hibernate.cache.use_query_cache" value="true"/>
      <property name="objectgrid.configuration" value="ObjectGridName=myOGName,ObjectGridType=
               EMBEDDED,MaxNumberOfReplicas=4" />
      <property name="objectgrid.hibernate.regionNames" value="queryRegion1, queryRegion2" />
    </properties>
</persistence-unit>
```

hibernate.cfg.xml

```
<hibernate-configuration>
    <session-factory>
        <!-- Database connection settings -->
        <property name="connection.driver_class">org.apache.derby.jdbc.EmbeddedDriver</property>
        <property name="connection.url">jdbc:derby:DB_testPU2;create=true</property>
        <!-- ObjectGrid cache setting-->
        <property name="cache.provider_class">com.ibm.websphere.objectgrid.hibernate.cache.
                 ObjectGridHibernateCacheProvider</property>
        <property name="cache.use_query_cache">true</property>
        <property name="objectgrid.configuration">ObjectGridName=myOGName,
                 ObjectGridType=EMBEDDED,MaxNumberOfReplicas=4 </property>
        <property name="objectgrid.hibernate.regionNames">queryRegion1, queryRegion2</property>

        <mapping resource="com/ibm/websphere/objectgrid/jpa/test/Employee.hbm.xml"/>

    </session-factory>
</hibernate-configuration>
```

Daten vorab in den ObjectGrid-Cache laden

Sie können die Methode ″preload″ der Klasse ″ObjectGridHibernateCacheProvider″ verwenden, um Daten für eine Entitätsklasse vorab in den ObjectGrid-Cache zu laden.

Beispiel 1

Mit EntityManagerFactory

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("testPU");
ObjectGridHibernateCacheProvider.preload("objectGridName", emf, TargetEntity.class, 100, 100);
```

Beispiel 2

Mit SessionFactory

```
org.hibernate.cfg.Configuration cfg = new Configuration();
// Methoden addResource, addClass, und setProperty von Configuration verwenden,
// um erforderliche Konfiguration zum Erstellen von SessionFactor vorzubereiten
SessionFactory sessionFactory= cfg.buildSessionFactory();
ObjectGridHibernateCacheProvider.preload("objectGridName", sessionFactory, TargetEntity.class, 100, 100);
```

Anmerkung:

1. In einem verteilten System kann dieser Preload-Mechanismus nur über eine einzige Java Virtual Machine aufgerufen werden. Der Preload-Mechanismus kann nicht gleichzeitig über mehrere JVMs ausgeführt werden.
2. Vor der Ausführung des Preload-Mechanismus müssen Sie den eXtreme-Scale-Cache initialisieren, indem Sie einen EntityManager über die EntityManagerFactory erstellen, damit alle entsprechenden BackingMaps erstellt werden. Andernfalls zwingt der Preload-Mechanismus die Initialisierung des Caches mit einer einzigen Standard-BackingMap für die Unterstützung aller Entitäten. Das bedeutet, dass eine einzige BackingMap von allen Entitäten gemeinsam genutzt wird.

Hibernate-Cachekonfiguration mit XML anpassen

Für die meisten Szenarios reicht die Definition von Cacheeigenschaften aus. Wenn Sie das vom Cache verwendete ObjectGrid weiter anpassen möchten, können Sie Hibernate-ObjectGrid-XML-Konfigurationsdateien im Verzeichnis `META-INF` bereitstellen, wie z. B. die Datei `persistence.xml`. Während der Initialisierung versucht der Cache, diese XML-Dateien zu finden und sie dann zu verarbeiten.

Es gibt drei Typen von Hibernate-ObjectGrid-XML-Konfigurationsdateien: `hibernate-objectGrid.xml` (ObjectGrid-Konfiguration), `hibernate-objectGridDeployment.xml` (Implementierungsrichtlinie) und `hibernate-objectGrid-client-override.xml` (Konfiguration der ObjectGrid-Clientkorrekturwerte). Je nach konfigurierter eXtreme-Scale-Topologie können Sie jede dieser drei XML-Dateien verwenden, um diese Topologie anzupassen.

Für die Typen EMBEDDED und EMBEDDED_PARTITION können Sie jede der drei XML-Dateien für die Anpassung des ObjectGrids, der Implementierungsrichtlinie oder der Konfiguration der ObjectGrid-Clientkorrekturwerte verwenden.

Bei ObjectGrids des Typs REMOTE erstellt der Cache kein dynamisches ObjectGrid. Vielmehr ruft der Cache ein clientseitiges ObjectGrid vom Katalogservice ab. Zum Anpassen der Konfiguration für die ObjectGrid-Clientkorrekturwerte können Sie nur eine Datei `hibernate-objectGrid-client-override.xml` verwenden.

1. **ObjectGrid-Konfiguration**: Verwenden Sie die Datei `META-INF/hibernate-objectGrid.xml`. Diese Datei wird verwendet, um die ObjectGrid-Konfiguration für die Typen EMBEDDED und EMBEDDED_PARTITION anzupassen. Für den Typ REMOTE wird diese Datei ignoriert. Standardmäßig hat jede Entitätsklasse einen zugeordneten Regionsnamen (standardmäßig den Namen der Entitätsklasse), der einer BackingMap-Konfiguration zugeordnet ist, die nach dem Regionsnamen in der ObjectGrid-Konfiguration benannt ist. Die Entitätsklasse ″com.mycompany.Employee″ hat beispielsweise standardmäßig einen zugeordneten Regionsnamen, der der BackingMap ″com.mycompany.Employee″ zugeordnet ist. Die BackingMap-Standardkonfiguration hat die Einstellungen readOnly=″false″, copyKey=″false″, lockStrategy=″NONE″ und `copyMode="NO_COPY"`. Sie können einige BackingMaps mit einer ausgewählten Konfiguration anpassen. Das reservierte Schlüsselwort ″ALL_ENTITY_MAPS″ kann verwendet werden, um alle Maps mit Ausnahme der angepassten Maps aus der Datei `hibernate-objectGrid.xml` darzustellen. BackingMaps, die nicht in der Datei `hibernate-objectGrid.xml` aufgelistet sind, verwenden die Standardkonfiguration.
2. **ObjectGridDeployment-Konfiguration**: Verwenden Sie die Datei `META-INF/hibernate-objectGridDeployment.xml`. Diese Datei wird verwendet, um die Implementierungsrichtlinie anzupassen. Wenn Sie bei der Anpassung der Implementierungsrichtlinie die Datei `hibernate-objectGridDeployment.xml`

bereitstellen, wird die Standardimplementierungsrichtlinie verworfen. Alle Attributwerte für die Implementierungsrichtlinie werden der bereitgestellten Datei `hibernate-objectGridDeployment.xml` entnommen.

3. **Konfiguration der ObjectGrid-Clientkorrekturwerte**: Verwenden Sie die Datei `META-INF/hibernate-objectGrid-client-override.xml`. Diese Datei wird verwendet, um ein clientseitiges ObjectGrid anzupassen. Standardmäßig wendet der ObjectGrid-Cache eine Standardkonfiguration für Clientkorrekturwerte an, die den nahen Cache inaktiviert. Wenn eine Anwendung einen nahen Cache erfordert, können Sie diese Datei bereitstellen und `numberOfBuckets="xxx"` angeben. Der Standardclientkorrekturwert inaktiviert den nahen Cache mit `numberOfBuckets="0"`. Der nahe Cache kann aktiv sein, wenn das Attribut "numberOfBuckets" über die Datei `hibernate-objectGrid-client-override.xml` auf einen Wert größer als 0 zurückgesetzt wird. Die Funktionsweise der Datei `hibernate-objectGrid-client-override.xml` gleicht der der Datei `hibernate-objectGrid.xml`: Sie überscheibt oder erweitert die Standardkonfiguration für ObjectGrid-Clientkorrekturwerte.

Beispiele für Hibernate-ObjectGrid-XML-Dateien

Hibernate-ObjectGrid-XML-Dateien müssen auf der Basis einer Persistenzeinheit erstellt werden.

Im Folgenden sehen Sie eine Beispieldatei `persistence.xml`, die die Konfiguration einer Persistenzeinheit darstellt:

persistence.xml

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  version="1.0">
  <persistence-unit name="AnnuityGrid">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>

    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.FixedAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.EquityAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Person</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityHolder</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Address</class>

    <exclude-unlisted-classes>true</exclude-unlisted-classes>

    <properties>
      <property name="hibernate.show_sql" value="false" />
      <property name="hibernate.connection.url" value="jdbc:db2:Annuity" />
      <property name="hibernate.connection.driver_class" value="com.ibm.db2.jcc.DB2Driver" />
      <property name="hibernate.default_schema" value="EJB30" />

      <!-- Cache -->
      <property name="hibernate.cache.provider_class"
        value="com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider" />
      <property name="hibernate.cache.use_query_cache" value="true" />
      <property name="objectgrid.configuration" value="ObjectGridType=EMBEDDED,
               ObjectGridName=Annuity, MaxNumberOfReplicas=4" />
    </properties>
  </persistence-unit>

</persistence>
```

Im Folgenden sehen Sie die Datei `hibernate-objectGrid.xml`, die der Datei `persistence.xml` entspricht:

hibernate-objectGrid.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="Annuity">
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" readOnly="false" copyKey="false"
```

```
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <backingMap name="defaultCacheMap" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="defaultCacheMap" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
      <backingMap name="org.hibernate.cache.UpdateTimestampsCache" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="org.hibernate.cache.UpdateTimestampsCache" />
      <backingMap name="org.hibernate.cache.StandardQueryCache" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="org.hibernate.cache.StandardQueryCache" />
    </objectGrid>
  </objectGrids>
  <backingMapPluginCollections>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="defaultCacheMap">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="org.hibernate.cache.UpdateTimestampsCache">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>


    </backingMapPluginCollection>
    <backingMapPluginCollection id="org.hibernate.cache.StandardQueryCache">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Anmerkung: Die Maps ″org.hibernate.cache.UpdateTimestampsCache″, ″org.hibernate.cache.StandardQueryCache″ und ″defaultCacheMap″ sind erforderlich.

Es folgt die Datei `hibernate-objectGridDeployment.xml`, die der Datei `persistence.xml` entspricht:

hibernate-objectGridDeployment.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="Annuity">
    <mapSet name="MAPSET_Annuity" numberOfPartitions="1" numInitialContainers="1" minSyncReplicas="0"
          maxSyncReplicas="4" maxAsyncReplicas="0" replicaReadEnabled="true">
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <map ref="defaultCacheMap" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
```

```
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
      <map ref="org.hibernate.cache.UpdateTimestampsCache" />
      <map ref="org.hibernate.cache.StandardQueryCache" />
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

Anmerkung: Die Maps ″org.hibernate.cache.UpdateTimestampsCache″, ″org.hibernate.cache.StandardQueryCache″ und ″defaultCacheMap″ sind erforderlich.

Externes System für einen Cache mit dem ObjectGrid-Typ REMOTE

Sie müssen ein externes eXtreme-Scale-System einrichten, wenn Sie einen Cache mit dem ObjectGrid-Typ REMOTE konfigurieren möchten. Sie benötigen ObjectGrid- und ObjectGridDeployment-XML-Konfigurationsdateien, die auf der Datei `persistence.xml` basieren, um ein externes System einrichten zu können. Die Hibernate-ObjectGrid- und ObjectGridDeployment-XML-Konfigurationsdateien, die im Abschnitt mit den Beispielen für Hibernate-ObjectGrid-XML-Dateien beschrieben werden, können auch zum Einrichten eines externen eXtreme-Scale-Systems verwendet werden.

Ein externes ObjectGrid-System hat sowohl Katalogservice- als auch ObjectGrid-Serverprozesse. Sie müssen einen Katalogservice starten, bevor Sie Containerserver starten. Weitere Informationen finden Sie in den Einzelheiten zum Starten eigenständiger Server von eXtreme Scale und Containerprozessen im *Administratorhandbuch*.

Fehlerbehebung

1. **CacheException: Failed to get ObjectGrid server**

   Bei einem ObjectGrid des Typs EMBEDDED oder EMBEDDED_PARTITION versucht der Cache, eine Serverinstanz von der Laufzeitumgebung von eXtreme Scale abzurufen. In einer Java-SE-Umgebung wird ein Server von eXtreme Scale mit integriertem Katalogservice gestartet. Der integrierte Katalogservice versucht, an Port 2809 empfangsbereit zu sein. Wenn dieser Port von einem anderen Prozess verwendet wird, tritt dieser Fehler auf. Wenn externe Katalogserviceendpunkte angegeben werden, z. B. in der Datei `objectGridServer.properties`, tritt dieser Fehler auf, wenn der Hostname oder Port falsch angegeben sind.

2. **CacheException: Failed to get REMOTE ObjectGrid for configured REMOTE ObjectGrid. objectGridName = [ObjectGridName], PU name = [persistenceUnitName]**

   Dieser Fehler tritt auf, wenn der Cache kein ObjectGrid von den bereitgestellten Katalogserviceendpunkten abrufen kann. Gewöhnlich ist dieser Fehler auf einen falsch angegebenen Hostnamen oder Port zurückzuführen.

3. **CacheException: Cannot have two PUs [persistenceUnitName_1, persistenceUnitName_2] configured with same ObjectGridName [ObjectGridName] of EMBEDDED ObjectGridType**

   Diese Ausnahme tritt ein, wenn Sie eine Konfiguration mit vielen Persistenzeinheiten haben und die Caches dieser Einheiten mit demselben ObjectGrid-Namen und dem ObjectGrid-Typ EMBEDDED konfiguriert sind. Diese Persistenzeinheitenkonfigurationen können in derselben oder in unterschiedlichen Dateien `persistence.xml` enthalten sein. Sie müssen sicherstellen, dass der ObjectGrid-Name für jede Persistenzeinheit eindeutig ist, wenn der ObjectGrid-Typ EMBEDDED verwendet wird.

4. **CacheException: REMOTE ObjectGrid [ObjectGridName] does not include required BackingMaps [mapName_1, mapName_2,...]**

   Wenn der ObjectGrid-Typ REMOTE verwendet wird und das abgerufene clientseitige ObjectGrid keine vollständigen Entitäts-BackingMaps für die Unterstützung des Caches der Persistenzeinheit hat, wird diese Ausnahme ausgelöst. Beispiel: Es sind fünf Entitätsklassen in der Konfiguration der Persistenzeinheit aufgelistet, aber das abgerufene ObjectGrid hat nur zwei BackingMaps. Diese Ausnahme wird auch dann ausgelöst, wenn das abgerufene ObjectGrid zehn BackingMaps enthält, aber eine der fünf erforderlichen Entitäts-BackingMaps nicht unter den zehn vorhandenen gefunden wird.

Konfiguration des OpenJPA-Cache-Plug-ins

WebSphere eXtreme Scale stellt DataCache- und QueryCache-Implementierungen für OpenJPA bereit. OpenJPA-ObjectGrid-Cache und OpenJPA-ObjectGrid-Cache sind allgemeine Kurzbezeichnungen für die DataCache- und QueryCache-Implementierungen.

Einstellungen

Der eXtreme-Scale-Cache wird für OpenJPA über die Konfigurationseigenschaften ″openjpa.DataCache″ und ″openjpa.QueryCache″ in der Datei `persistence.xml` aktiviert und inaktiviert. Die Syntax für die Einstellung der Eigenschaften ist wie folgt:

```
<property name="openjpa.DataCache"
          value="<object_grid_datacache_class(<Eigenschaft>=<Wert>,...)"/>
<property name="openjpa.QueryCache"
          value="<object_grid_querycache_class(<Eigenschaft>=<Wert>,...)"/>
```

DataCache und QueryCache können die Eigenschaften des eXtreme-Scale-Caches verwenden, um den von der Persistenzeinheit verwendeten Cache zu konfigurieren.

Zusätzlich zur Einstellung für den eXtreme-Scale-Cache muss die Eigenschaft ″openjpa.RemoteCommitProvider″ auf `sjvm` gesetzt werden:

```
<property name="openjpa.RemoteCommitProvider" value="sjvm"/>
```

Das mit der Annotation ″@DataCache″ für jede Entitätsklasse angegebene Zeitlimit wird an die BackingMap weitergegeben, in der jede Entität zwischengespeichert wird. Der mit der Annotation ″@DataCache″ angegebene Namenswert wird jedoch vom eXtreme-Scale-Cache ignoriert. Der vollständig qualifizierte Entitätsklassenname ist der Name der Cache-Map.

Die Methoden ″pin″ und ″unpin″ von OpenJPA-StoreCache und OpenJPA-QueryCache werden nicht unterstützt und haben keine Funktion.

Sie können die Eigenschaft ″ObjectGridName″, die Eigenschaft ″ObjectGridType″ und andere einfache auf die Implementierungsrichtlinie bezogene Eigenschaften in der Eigenschaftenliste der ObjectGrid-Cacheklasse angeben, um die Cachekonfiguration anzupassen. Es folgt ein Beispiel:

```
<property name="openjpa.DataCache"
          value="com.ibm.websphere.objectgrid.openjpa.ObjectGridDataCache(
          ObjectGridName=BasicTestObjectGrid,ObjectGridType=EMBEDDED,
          maxNumberOfReplicas=4)"/>
<property name="openjpa.QueryCache"
          value="com.ibm.websphere.objectgrid.openjpa.ObjectGridQueryCache()"/>
<property name="openjpa.RemoteCommitProvider" value="sjvm"/>
```

DataCache- und QueryCache-Konfigurationen sind voneinander unabhängig. Sie können beide Konfigurationen aktivieren. Wenn Sie jedoch beide Konfigurationen aktivieren, verwendet QueryCache dieselbe Konfiguration wie DataCache, und die Konfiguration von QueryCache wird verworfen.

OpenJPA-Cachekonfiguration mit XML anpassen

Für die meisten Szenarios ist die Definition der Eigenschaften des eXtreme-Scale-Caches ausreichend. Um das vom Cache verwendete ObjectGrid weiter anzupassen, können Sie ObjectGrid-XML-Konfigurationsdateien für OpenJPA wie die Datei `persistence.xml` in Ihrem Verzeichnis META-INF bereitstellen. Während der Cacheinitialisierung versucht der ObjectGrid-Cache, diese XML-Datei zu finden und dann zu verarbeiten.

Es gibt drei Typen von OpenJPA-ObjectGrid-XML-Konfigurationsdateien: `openjpa-objectGrid.xml` (ObjectGrid-Konfiguration), `openjpa-objectGridDeployment.xml` (Implementierungsrichtlinie) und `openjpa-objectGrid-client-override.xml` (Konfiguration der ObjectGrid-Clientkorrekturwerte). Je nach konfiguriertem ObjectGrid-Typ können Sie jede dieser Datei XML-Dateien verwenden, um das ObjectGrid anzupassen.

Für die Typen EMBEDDED und EMBEDDED_PARTITION können Sie jede der drei XML-Dateien für die Anpassung des ObjectGrids, der Implementierungsrichtlinie oder der Konfiguration der ObjectGrid-Clientkorrekturwerte verwenden.

Bei ObjectGrids des Typs REMOTE erstellt der ObjectGrid-Cache kein dynamisches ObjectGrid. Vielmehr ruft der Cache nur ein clientseitiges ObjectGrid vom Katalogservice ab. Zum Anpassen der Konfiguration für die ObjectGrid-Clientkorrekturwerte können Sie nur eine Datei `openjpa-objectGrid-client-override.xml` verwenden.

1. **ObjectGrid-Konfiguration**: Verwenden Sie die Datei `META-INF/openjpa-objectGrid.xml`. Diese Datei wird verwendet, um die ObjectGrid-Konfiguration für die Typen EMBEDDED und EMBEDDED_PARTITION anzupassen. Für den Typ REMOTE wird diese Datei ignoriert. Standardmäßig wird jede Entitätsklasse einer eigenen BackingMap-Konfiguration zugeordnet, die denselben Namen wie die Entitätsklasse in der ObjectGrid-Konfiguration hat. Die Entitätsklasse "com.mycompany.Employee" wird beispielsweise der BackingMap "com.mycompany.Employee" zugeordnet. Die BackingMap-Standardkonfiguration hat die Einstellungen `readOnly="false"`, `copyKey="false"`, `lockStrategy="NONE"` und `copyMode="NO_COPY"`. Sie können einige BackingMaps mit der ausgewählten Konfiguration anpassen. Sie können das reservierte Schlüsselwort `ALL_ENTITY_MAPS` verwenden, um alle Maps darzustellen mit Ausnahme anderer angepasster Maps, die in der Datei `openjpa-objectGrid.xml` aufgelistet sind. BackingMaps, die nicht in der Datei `openjpa-objectGrid.xml` aufgelistet sind, verwenden die Standardkonfiguration. Wenn für angepasste BackingMaps kein Attribut "BackingMaps" oder keine Eigenschaften definiert sind, diese Attribute aber in der Standardkonfiguration angegeben sind, werden die Attributwerte aus der Standardkonfiguration angewendet. Ist eine Entitätsklasse beispielsweise mit `timeToLive=30` annotiert, enthält die BackingMap-Standardkonfiguration für diese Entität ebenfalls `timeToLive=30`. Wenn die angepasste Datei `openjpa-objectGrid.xml` diese BackingMap ebenfalls enthält, aber den timeToLive-Wert nicht definiert, wird für die angepasste BackingMap standardmäßig der Wert `timeToLive=30` verwendet. Die Datei `openjpa-objectGrid.xml` soll die Standardkonfiguration überschreiben oder erweitern.

2. **ObjectGridDeployment-Konfiguration**: Verwenden Sie die Datei `META-INF/openjpa-objectGridDeployment.xml`. Diese Datei wird verwendet, um die Implementierungsrichtlinie anzupassen. Wenn Sie bei der Anpassung der Implementierungsrichtlinie die Datei `openjpa-objectGridDeployment.xml` bereitstellen, wird die Standardimplementierungsrichtlinie verworfen. Alle Attributwerte für die Implementierungsrichtlinie stammen aus der bereitgestellten Datei `penjpa-objectGridDeployment.xml`.
3. **Konfiguration der ObjectGrid-Clientkorrekturwerte**: Verwenden Sie die Datei `META-INF/openjpa-objectGrid-client-override.xml`. Diese Datei wird verwendet, um ein clientseitiges ObjectGrid anzupassen. Standardmäßig wendet der ObjectGrid-Cache eine ObjectGrid-Standardkonfiguration für Clientkorrekturwerte an, die einen nahen Cache inaktiviert. Wenn eine Anwendung einen nahen Cache erfordert, können Sie diese Datei bereitstellen und `numberOfBuckets="xxx"` angeben. Der Standardclientkorrekturwert inaktiviert den nahen Cache mit `numberOfBuckets="0"`. Der nahe Cache kann aktiv sein, wenn das Attribut "numberOfBuckets" über die Datei `openjpa-objectGrid-client-override.xml` auf einen Wert größer als 0 zurückgesetzt wird. Die Datei `openjpa-objectGrid-client-override.xml` funktioniert ähnlich wie die Datei `openjpa-objectGrid.xml`. Sie überscheibt oder erweitert die Standardkonfiguration für ObjectGrid-Clientkorrekturwerte.

Beispiele für OpenJPA-ObjectGrid-XML-Dateien

OpenJPA-ObjectGrid-XML-Dateien müssen auf der Basis der Persistenzeinheit erstellt werden.

Im Folgenden sehen Sie eine Beispieldatei `persistence.xml`, die die Konfiguration einer Persistenzeinheit darstellt:

persistence.xml

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  version="1.0">
  <persistence-unit name="AnnuityGrid">
    <provider>org.apache.openjpa.persistence.PersistenceProviderImpl</provider>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.FixedAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.EquityAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Person</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityHolder</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Address</class>
    <exclude-unlisted-classes>true</exclude-unlisted-classes>

    <properties>
    <!-- Database setting -->


    <!--  enable cache -->
      <property name="openjpa.DataCache"
        value="com.ibm.websphere.objectgrid.openjpa.ObjectGridDataCache(objectGridName=Annuity,
               objectGridType=EMBEDDED, maxNumberOfReplicas=4)" />
      <property name="openjpa.RemoteCommitProvider" value="sjvm" />
      <property name="openjpa.QueryCache"
                value="com.ibm.websphere.objectgrid.openjpa.ObjectGridQueryCache()" />
    </properties>
  </persistence-unit>
</persistence>
```

Es folgt die Datei `openjpa-objectGrid.xml`, die der Datei `persistence.xml` entspricht:

openjpa-objectGrid.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
xmlns="http://ibm.com/ws/objectgrid/config">
<objectGrids>
  <objectGrid name="Annuity">
    <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
    <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" />
    <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
    <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
    <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
    <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject"
                readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject" />
    <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
    <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
    <backingMap name="ObjectGridQueryCache" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" pluginCollectionRef="ObjectGridQueryCache"
                evictionTriggers="MEMORY_USAGE_THRESHOLD"  />
  </objectGrid>
</objectGrids>
<backingMapPluginCollections>
  <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity">
    <bean id="ObjectTransformer"
          className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
    </bean>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address">
    <bean id="ObjectTransformer"
          className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
    </bean>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor">
    <bean id="ObjectTransformer"
          className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
    </bean>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person">
    <bean id="ObjectTransformer"
          className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
    </bean>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact">
    <bean id="ObjectTransformer"
          className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
    </bean>
  </backingMapPluginCollection>
  <backingMapPluginCollection
          id="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject">
    <bean id="ObjectTransformer"
          className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
    </bean>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider">
    <bean id="ObjectTransformer"
          className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
    </bean>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout">
    <bean id="ObjectTransformer"
          className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
    </bean>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="ObjectGridQueryCache">
    <bean id="MapIndexPlugin"  className="com.ibm.websphere.objectgrid.plugins.index.HashIndex">
        <property name="Name" type="java.lang.String"
                  value="QueryCacheKeyIndex" description="name of index"/>
        <property name="POJOKeyIndex" type="boolean" value="true" description="POJO Key Index "/>
    </bean>
    <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
```

```
      </bean>
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Anmerkung:

1. Jede Entität wird einer BackingMap zugeordnet, die denselben Namen wie die vollständig qualifizierte Entitätsklasse hat.
2. Wenn Entitätsklassen in einer Vererbungshierarchie enthalten sind, werden untergeordnete Klassen der BackingMap der übergeordneten Klasse zugeordnet. Das bedeutet, dass eine Vererbungshierarchie eine einzige BackingMap nutzt.
3. Die ObjectGridQueryCache-Map ist für die Unterstützung von QueryCache erforderlich.
4. Die backingMapPluginCollection für jede Entitäts-Map erfordert, dass der ObjectTransformer die Klasse ″com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer″ verwendet.
5. Die backingMapPluginCollection für ObjectGridQueryCache muss einen Schlüsselindex mit dem Namen ″QueryCacheKeyIndex″ haben, wie im folgenden Beispiel gezeigt wird.
6. Das Bereinigungsprogramm (Evictor) ist für jede Map optional.

Es folgt die Datei `openjpa-objectGridDeployment.xml`, die der Datei `persistence.xml` entspricht:

openjpa-objectGridDeployment.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="Annuity">
    <mapSet name="MAPSET_Annuity" numberOfPartitions="1" numInitialContainers="1"
            minSyncReplicas="0" maxSyncReplicas="4" maxAsyncReplicas="0"
            replicaReadEnabled="true">
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
      <map ref="ObjectGridQueryCache" />
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

Anmerkung: Die ObjectGridQueryCache-Map ist für die Unterstützung von QueryCache erforderlich.

Externes System für einen Cache mit dem ObjectGrid-Typ REMOTE

Sie müssen ein externes System einrichten, wenn Sie einen Cache mit dem ObjectGrid-Typ REMOTE konfigurieren möchten. Sie benötigen ObjectGrid- und ObjectGridDeployment-XML-Konfigurationsdateien, die auf einer Datei `persistence.xml` basieren, um ein externes System einrichten zu können. Die OpenJPA-ObjectGrid- und -ObjectGridDeployment-XML-Konfigurationsdateien, die im Abschnitt mit den Beispielen für die OpenJPA-ObjectGrid-XML-Dateien beschrieben werden, können auch zum Einrichten eines externen eXtreme-Scale-Systems verwendet werden.

Ein externes eXtreme-Scale-System hat sowohl Katalogservice- als auch Containerserverprozesse. Sie müssen den Katalogserver starten, bevor Sie die Containerserver starten.

Fehlerbehebung

1. **CacheException: Failed to get ObjectGrid server**

   Bei einem ObjectGrid des Typs EMBEDDED oder EMBEDDED_PARTITION versucht der eXtreme-Scale-Cache, eine Serverinstanz von der Laufzeitumgebung abzurufen. In einer Java-SE-Umgebung wird ein Server von eXtreme Scale mit integriertem Katalogservice gestartet. Der integrierte Katalogservice versucht, an Port 2809 empfangsbereit zu sein. Wenn dieser Port von einem anderen Prozess verwendet wird, tritt dieser Fehler auf. Wenn externe Katalogserviceendpunkte angegeben werden, z. B. in der Datei `objectGridServer.properties`, tritt dieser Fehler auf, wenn der Hostname oder Port falsch angegeben sind.

2. **CacheException: Failed to get REMOTE ObjectGrid for configured REMOTE ObjectGrid. objectGridName = [ObjectGridName], PU name = [persistenceUnitName]**

   Dieser Fehler tritt auf, wenn der Cache kein ObjectGrid von den bereitgestellten Katalogserviceendpunkten abrufen kann. Gewöhnlich ist dieser Fehler auf einen falsch angegebenen Hostnamen oder Port zurückzuführen.

3. **CacheException: Cannot have two PUs [persistenceUnitName_1, persistenceUnitName_2] configured with same ObjectGridName [ObjectGridName] of EMBEDDED ObjectGridType**

   Diese Ausnahme tritt ein, wenn Sie viele Persistenzeinheitenkonfigurationen haben und die eXtreme-Scale-Caches dieser Einheiten mit demselben ObjectGrid-Namen und dem ObjectGrid-Typ EMBEDDED konfiguriert sind. Diese Persistenzeinheitenkonfigurationen können in derselben oder in unterschiedlichen Dateien `persistence.xml` enthalten sein. Sie müssen sicherstellen, dass der ObjectGrid-Name für jede Persistenzeinheit eindeutig ist, wenn der ObjectGrid-Typ EMBEDDED verwendet wird.

4. **CacheException: REMOTE ObjectGrid [ObjectGridName] does not include required BackingMaps [mapName_1, mapName_2,...]**

   Wenn der ObjectGrid-Typ REMOTE verwendet wird und das abgerufene clientseitige ObjectGrid keine vollständigen Entitäts-BackingMaps für die Unterstützung des Caches der Persistenzeinheit hat, wird diese Ausnahme ausgelöst. Beispiel: Es sind fünf Entitätsklassen in der Konfiguration der Persistenzeinheit aufgelistet, aber das abgerufene ObjectGrid hat nur zwei BackingMaps. Diese Ausnahme wird auch dann ausgelöst, wenn das abgerufene ObjectGrid zehn BackingMaps enthält, aber eine der fünf erforderlichen Entitäts-BackingMaps nicht unter den zehn vorhandenen gefunden wird.

Anmerkung: Das Datenformat des eXtreme-Scale-Caches für OpenJPA wurde geändert, um eine bessere Leistung zu erzielen. Alle Systeme, auf denen OpenJPA-Anwendungen ausgeführt werden, die mit eXtreme Scale als L2-Cache konfiguriert sind, müssen gestoppt werden, bevor die Migration auf WebSphere eXtreme Scale Version 7.0 durchgeführt wird.

Unterstützung des Write-Behind-Cachings

Sie können Write-Behind-Caching verwenden, um die Kosten bei der Aktualisierung einer Back-End-Datenbank zu reduzieren. Beim Write-Behind-Caching werden Aktualisierungen für das Ladeprogramm-Plug-in in die Warteschlange eingereiht.

Einführung

Beim Write-Behind-Caching werden Aktualisierungen für das Ladeprogramm-Plug-in asynchron in die Warteschlange eingereiht. Sie können die Leistung von

Aktualisierungs-, Einfüge- und Entfernungsoperationen für die Map verbessern, indem Sie die eXtreme-Scale-Transaktion von der Datenbanktransaktion entkoppeln. Die asynchrone Aktualisierung wird nach einer zeitbasierten Verzögerung (z. B. fünf Minuten) oder einer eintragsbasierten Verzögerung (z. B. 1000 Einträge) durchgeführt.

Wenn Sie die Write-Behind-Einstellung in einer BackingMap konfigurieren, wird ein Write-Behind-Thread erstellt und der konfigurierte Loader eingeschlossen. Wenn eine eXtreme-Scale-Transaktion einen Eintrag in einer eXtreme-Scale-Map einfügt, aktualisiert oder entfernt, wird ein LogElement-Objekt für jeden dieser Datensätze erstellt. Diese Elemente werden an das Write-Behind-Ladeprogramm gesendet und in eine spezielle ObjectMap, eine so genannte Warteschlangen-Map, eingereiht. Jede BackingMap mit aktivierter Write-Behind-Einstellung hat ihre eigenen Warteschlangen-Maps. Ein Write-Behind-Thread entfernt die in die Warteschlange eingereihten Daten aus den Warteschlangen-Maps und überträgt Sie mit Push in das echte Back-End-Ladeprogramm.

Das Write-Behind-Ladeprogramm sendet nur LogElement-Objekte der Typen ″insert″ (Einfügen), ″update″ (Aktualisieren) und ″delete″ (Löschen) an das echte Ladeprogramm. Alle anderen Typen von LogElement-Objekten, wie z. B. EVICT, werden ignoriert.

Vorteile

Das Aktivieren der Write-Behind-Unterstützung hat die folgenden Vorteile:

- Isolation von Back-End-Fehlern: Durch das Write-Behind-Caching können Back-End-Fehler isoliert werden. Wenn die Back-End-Datenbank ausfällt, werden Aktualisierungen in die Warteschlangen-Map eingereiht. Die Anwendungen können weiterhin Transaktionen an eXtreme Scale senden. Nach der Wiederherstellung des Back-Ends werden die Daten in der Warteschlangen-Map mit Push an das Back-End übertragen.
- Geringere Back-End-Last: Das Write-Behind-Ladeprogramm fasst die Aktualisierungen auf Schlüsselbasis so zusammen, dass nur eine einzige zusammenfasste Aktualisierung pro Schlüssel in der Warteschlangen-Map vorhanden ist. Bei dieser Zusammenfassung verringert sich die Anzahl der Aktualisierungen für das Back-End.
- Verbesserte Transaktionsleistung: Die Zeiten einzelner eXtreme-Scale-Transaktionen verringern sich, weil sie nicht auf die Synchronisation der Daten mit dem Back-End-Warten müssen.

ObjectGrid-XML-Deskriptordatei

Wenn Sie eXtreme Scale mit einer eXtreme-Scale-XML-Deskriptordatei konfigurieren, wird der Write-Behind-Loader über das Attribut ″writeBehind″ im Tag ″backingMap″ aktiviert. Es folgt ein Beispiel:

```
<objectGrid name="library" >
     <backingMap name="book" writeBehind="T300;C900" pluginCollectionRef="bookPlugins"/>
```

Im vorherigen Beispiel wird die Write-Behind-Unterstützung für die BackingMap ″book″ mit dem Parameter ″T300;C900″ aktiviert.

Das Attribut ″writeBehind″ gibt die maximale Aktualisierungszeit und/oder eine maximale Anzahl an Schlüsselaktualisierungen an. Der Parameter ″write-behind″ hat das folgende Format:

```
Attribut "writeBehind" ::= <defaults> | <Aktualisierungszeit> | <Anzahl der Schlüsselaktualisierungen> | <Aktualisierungszeit> ";" <Anzahl de
Aktualisierungszeit  ::= "T" <positive ganze Zahl>
Anzahl der Schlüsselaktualisierungen  ::= "C" <positive ganze Zahl>
defaults  ::= "" {table}
```

Aktualisierungen im Loader finden statt, wenn eines der folgenden Ereignisse eintritt:

1. Die maximale Aktualisierungszeit in Sekunden seit der letzten Aktualisierung ist abgelaufen.
2. Die Anzahl aktualisierter Schlüssel in der Warteschlangen-Map hat die maximal zulässige Anzahl an Schlüsselaktualisierungen erreicht.

Diese Parameter sind lediglich Hinweise. Der echte Aktualisierungszähler und die echte Aktualisierungszeit liegen nah bei den Parametern. Es ist jedoch nicht garantiert, dass der echte Aktualisierungszähler und die echte Aktualisierungszeit den definierten Parametern entsprechen. Außerdem könnte die erste Write-Behind-Aktualisierung erst nach zwei Aktualisierungszeitintervallen stattfinden. Dies ist darauf zurückzuführen, dass eXtreme Scale die Startzeit für die Aktualisierung zufällig wählt, so dass nicht alle Partitionen gleichzeitig auf die Datenbank zugreifen.

Im vorherigen Beispiel (T300;C900) schreibt der Loader die Daten 300 Sekunden nach der letzten Aktualisierung bzw. bei 900 zu aktualisierenden Schlüsseln in die Datenbank.

Die Standardaktualisierungszeit sind 300 Sekunden, und die Standardanzahl der Schlüsselaktualisierungen ist 1000.

In der folgenden Tabelle sind einige Beispiele für Write-behind-Attribute aufgelistet.

Anmerkung: Wenn Sie den Write-Behind-Loader als leere Zeichenfolge konfigurieren (`writeBehind=""`), wird der Write-Behind-Loader mit den Standardwerten aktualisiert. Geben Sie das Attribut ″writeBehind″ nicht an, wenn die Write-Behind-Unterstützung nicht aktiviert werden soll.

Tabelle 6. Write-behind-Optionen

Attributwert	Zeit
T100	Die Aktualisierungszeit sind 100 Sekunden, und die Anzahl der Schlüsselaktualisierungen ist 1000 (Standardwert).
C2000	Die Aktualisierungszeit sind 300 Sekunden (Standardwert), und die Anzahl der Schlüsselaktualisierungen ist 2000.
T300;C900	Die Aktualisierungszeit sind 300 Sekunden, und die Anzahl der Schlüsselaktualisierungen ist 900.
″″	Die Aktualisierungszeit sind 300 Sekunden (Standardwert), und die Anzahl der Schlüsselaktualisierungen ist 1000 (Standardwert).

Write-Behind-Unterstützung programmgesteuert aktivieren

Wenn Sie eine BackingMap für eine lokale speicherinterne eXtreme-Scale-Instanz programmgesteuert erstellen, können Sie die folgende Methode in der Schnittstelle ″BackingMap″ verwenden, um die Write-Behind-Unterstützung zu aktivieren und zu inaktivieren.

```
public void setWriteBehind(String writeBehindParam);
```

Weitere Einzelheiten zur Verwendung der Methode ″setWriteBehind″ finden Sie im Abschnitt „Schnittstelle ″BackingMap″″ auf Seite 67.

Hinweise zum Anwendungsdesign

Das Aktivieren der Write-Behind-Unterstützung ist zwar einfach, aber eine Anwendung mit Write-Behind-Unterstützung zu entwerfen, bedarf sorgfältiger Überlegungen. Ohne Write-Behind-Unterstützung ist die Back-End-Transaktion in die eXtreme-Scale-Transaktion eingeschlossen. Die eXtreme-Scale-Transaktion wird vor der Back-End-Transaktion gestartet und endet erst nach Abschluss der Back-End-Transaktion.

Wenn die Write-Behind-Unterstützung aktiviert ist, endet die eXtreme-Scale-Transaktion vor dem Start der Back-End-Transaktion. Die eXtreme-Scale-Transaktion und die Back-End-Transaktion sind entkoppelt.

Referenzielle Integritätsbedingungen

Jede BackingMap, die mit Write-Behind-Unterstützung konfiguriert ist, hat einen eigenen Write-Behind-Thread, der die Daten mit Push an das Back-End überträgt. Deshalb werden die Daten, die in einer einzigen eXtreme-Scale-Transaktion in verschiedenen Maps aktualisiert wurden, im Back-End in verschiedenen Back-End-Transaktionen aktualisiert. Beispiel: Transaktion T1 aktualisiert den Schlüssel ″key1″ in der Map ″Map1″ und den Schlüssel ″key2″ in der Map ″Map2″. Die Aktualisierungen von Schlüssel key1 in der Map Map1 und von Schlüssel ″key2″ in der Map ″Map2″ werden in einer jeweils anderen Back-End-Transaktion von einem jeweils anderen Write-Behind-Thread durchgeführt. Wenn es Beziehungen zwischen den in Map1 und Map2 gespeicherten Daten, wie z. B. Integritätsbedingungen über Fremdschlüssel, im Back-End gibt, können die Aktualisierungen fehlschlagen.

Beim Design der referenziellen Integritätsbedingungen in Ihrer Back-End-Datenbank müssen Sie sicherstellen, dass solche nicht ausführbaren Aktualisierungen zugelassen werden.

Fehlgeschlagene Aktualisierungen

Da die eXtreme-Scale-Transaktion vor dem Start der Back-End-Transaktion beendet wird, ist es möglich, dass eine erfolgreiche Transaktion berichtet wird, obwohl dies eigentlich nicht der Fall ist. Wenn Sie beispielsweise versuchen, einen Eintrag in eine eXtreme-Scale-Transaktion einzufügen, die nicht in der BackingMap, aber im Back-End vorhanden ist führt dies zwar zu einem doppelten Schlüssel, aber die eXtreme-Scale-Transaktion ist trotzdem erfolgreich. Die Transaktion, in der der Write-Behind-Thread das Objekt in das Back-End einfügt, scheitert jedoch mit einer Ausnahme vom Typ ″Schlüssel doppelt vorhanden″.

Informationen zur Behandlung solcher Fehler finden Sie im Abschnitt „Behandlung fehlgeschlagener Write-Behind-Aktualisierungen” auf Seite 126.

Sperrverhalten von Warteschlangen-Maps

Ein weiterer wichtiger Unterschied im Transaktionsverhalten ist das Sperrverhalten. eXtreme Scale unterstützt drei verschiedene Sperrstrategien: PESSIMISTIC (Pessimistisch), OPTIMISITIC (Optimistisch) und NONE (Keine). Die Write-Behind-Warteschlangen-Map verwendet die pessimistische Sperrstrategie, unabhängig davon, welche Sperrstrategie für die zugehörige BackingMap konfiguriert ist. Es gibt zwei verschiedene Typen von Operationen, die eine Sperre für die Warteschlangen-Map anfordern:

- Wenn eine eXtreme-Scale-Transaktion festgeschrieben wird oder eine Flush-Operation (Map-Flush oder Sitzungs-Flush) stattfindet, liest die Transaktion den Schlüssel in der Warteschlangen-Map und setzt eine S-Sperre für den Schlüssel.
- Wenn eine eXtreme-Scale-Transaktion festgeschrieben wird, versucht die Transaktion die S-Sperre für den Schlüssel in eine X-Sperre zu aktualisieren.

Anhand dieses zusätzlichen Verhaltens für die Warteschlangen-Map sind einige Unterschiede im Sperrverhalten erkennbar.

- Wenn die Benutzer-Map mit einer pessimistischen Sperrstrategie konfiguriert ist, sind die Unterschiede im Sperrverhalten nicht gravierend. Bei jedem Aufruf einer Flush- oder Festschreiboperation (Commit) wird eine S-Sperre für denselben Schlüssel in der Warteschlangen-Map gesetzt. Während der Festschreibung wird nicht nur eine X-Sperre für den Schlüssel in der Benutzer-Map, sondern auch für den Schlüssel in der Warteschlangen-Map angefordert.
- Wenn die Benutzer-Map mit einer optimistischen Sperrstrategie oder ohne Sperrstrategie konfiguriert ist, folgt die Benutzertransaktion dem Muster der pessimistischen Sperrstrategie. Bei jedem Aufruf einer Flush- oder Festschreiboperation (Commit) wird eine S-Sperre für denselben Schlüssel in der Warteschlangen-Map angefordert. Während der Festschreibung wird in derselben Transaktion eine X-Sperre für den Schlüssel in der Warteschlangen-Map angefordert.

Transaktionswiederholungen im Ladeprogramm

WebSphere eXtreme Scale unterstützt keine zweiphasigen Transaktionen und keine XA-Transaktionen. Der Write-Behind-Thread entfernt Datensätze aus der Warteschlangen-Map und aktualisiert die Datensätze im Back-End. Wenn der Server mitten in der Transaktion ausfällt, können einige Back-End-Aktualisierungen verloren gehen.

Das Write-Behind-Ladeprogramm versucht automatisch, fehlgeschlagene Transaktionen erneut zu schreiben, und sendet eine unbestätigte Protokollfolge an das Back-End, um einen Datenverlust zu verhindern. Diese Aktion erfordert, dass das Ladeprogramm idempotent ist, d. h., wenn die Methode "Loader.batchUpdate(TxId, LogSequence)" zweimal mit demselben Wert aufgerufen wird, liefen diese Aufrufe dasselbe Ergebnis wie ein einmaliger Aufruf. Ladeprogrammimplementierungen müssen zum Aktivieren dieses Features die Schnittstelle "RetryableLoader" implementieren. Weitere Einzelheiten finden Sie in der API-Dokumentation.

Ausfall des Ladeprogramms

Das Ladeprogramm-Plug-in kann ausfallen, wenn es nicht mit dem Datenbank-Back-End kommunizieren kann. Dies kann passieren, wenn der Datenbankserver oder die Netzverbindung inaktiv ist. Das Write-Behind-Ladeprogramm reiht die Aktualisierungen in eine Warteschlange ein und versucht anschließend in regelmäßigen Abständen, die Datenänderungen mit Push an das Ladeprogramm zu übertragen. Das Ladeprogramm muss die Laufzeitumgebung von WebSphere eXtreme Scale darüber benachrichtigen, dass ein Problem mit der Datenbankkonnektivität vorliegt, indem es eine Ausnahme vom Typ "LoaderNotAvailableException" auslöst.

Deshalb muss die Ladeprogrammimplementierung in der Lage sein, einen Datenfehler von einem physischen Ausfall des Ladeprogramms zu unterscheiden. Bei Datenfehlern muss eine Ausnahme des Typs "LoaderException" oder "OptimisticCollisionException" ausgelöst bzw. erneut ausgelöst werden, aber beim physischen

Ausfall des Ladeprogramms muss eine Ausnahme des Typs ″LoaderNotAvailableException″ ausgelöst werden. WebSphere eXtreme Scale behandelt diese beiden Ausnahmen auf unterschiedliche Weise:

- Wenn das Write-Behind-Ladeprogramm eine Ausnahme vom Typ ″LoaderException″ abfängt, geht es von einem Datenfehler aus, z. B. von einem doppelten Schlüssel. Das Write-Behind-Ladeprogramm löst den Aktualisierungsstapel auf und versucht, einen Datensatz nach dem anderen zu aktualisieren, um den Datenfehler zu isolieren. Wird bei dieser Aktualisierung auf Datensatzbasis erneut eine Ausnahme vom Typ ″LoaderException″ abgefangen, wird ein Datensatz zur fehlgeschlagenen Aktualisierung erstellt und in der Map für fehlgeschlagene Aktualisierungen protokolliert.
- Wenn das Write-Behind-Ladeprogramm eine Ausnahme vom Typ ″LoaderNotAvailableException″ abfängt, geht es von einem Ausfall aus, weil es keine Verbindung zum Datenbank-Back-End herstellen kann, z. B., weil das Datenbank-Back-End inaktiv ist, keine Datenbankverbindung verfügbar ist oder das Netz inaktiv ist. Das Write-Behind-Ladeprogramm wartet 15 Sekunden und versucht dann erneut, die Datenbankaktualisierung im Stapelbetrieb durchzuführen.

Häufig wird der Fehler gemacht, eine Ausnahme vom Typ ″LoaderException″ auszulösen, obwohl eigentlich eine Ausnahme vom Typ ″LoaderNotAvailableException″ ausgelöst werden müsste. Alle Datensätze, die in die Warteschlange für das Write-Behind-Ladeprogramm eingereiht sind, werden als Datensätze für eine fehlgeschlagene Aktualisierung markiert, was den eigentlich Zweck der Isolierung von Back-End-Fehlern zunichte macht. Dieser Fehler ist wahrscheinlich, wenn Sie einen generischen Loader für die Kommunikation mit Datenbanken schreiben.

Der von eXtreme Scale bereitgestellte JPALoader ist ein Beispiel. Der JPALoader verwendet die JPA-API für die Interaktion mit Datenbank-Back-Ends. Wenn das Netz ausfällt, empfängt der JPALoader eine Ausnahme des Typs ″javax.persistence-.PersitenceException″, aber er weiß nichts über den Schweregrad des Fehlers, sofern der SQL-Status- und -Fehlercode der verketteten SQLException nicht geprüft werden. Die Tatsache, dass der JPALoader für alle Typen von Datenbanken verwendet werden kann, macht das Problem noch komplexer, da die SQL-Status und -Fehlercodes für Netzausfallprobleme anders sind. Für die Lösung dieses Problems stellt WebSphere eXtreme Scale eine API ″ExceptionMapper″ bereit, damit Benutzer eine Implementierung integrieren können, um eine Ausnahme einer aussagefähigeren Ausnahme zuordnen zu können. Benutzer können beispielsweise eine generische Ausnahme ″javax.persistence.PersitenceException″ einer Ausnahme ″LoaderNotAvailableException″ zuordnen, wenn der SQL-Status -bzw. -Fehlercode darauf hinweist, dass das Netz ausgefallen ist.

Leistungsaspekte

Die Unterstützung des Write-Behind-Cachings erhöht die Antwortzeiten, weil die Ladeprogrammaktualisierung aus der Transaktion entfernt wird. Außerdem erhöht sich der Datenbankdurchsatz, weil Datenbankaktualisierungen kombiniert werden. Es ist wichtig, die Kosten zu kennen, die durch den Write-Behind-Thread anfallen, der die Daten aus der Warteschlangen-Map extrahiert und mit Push an das Ladeprogramm überträgt.

Die maximale Aktualisierungsanzahl und die maximale Aktualisierungszeit müssen den erwarteten Verwendungsmustern und der Umgebung entsprechend angepasst werden. Wenn der Wert für die maximale Aktualisierungsanzahl oder der Wert für die maximale Aktualisierungszeit zu klein gewählt wird, kann der Write-Behind-Threads mehr Kosten verursachen, als er Vorteile bringt. Wenn ein sehr hoher Wert

für diese beiden Parameter festgelegt wird, ist es möglich, dass die Speicherbelegung aufgrund der Einreihung der Daten zunimmt und veraltete Datensätze länger in der Datenbank verbleiben.

Um die beste Leistung zu erzielen, sollten Sie bei der Optimierung der Write-Behind-Parameter die folgenden Faktoren berücksichtigen:

- Verhältnis zwischen Lese- und Schreibtransaktionen
- Aktualisierungsintervall für dieselben Datensätze
- Latenzzeit für Datenbankaktualisierung

Write-Behind-Unterstützung konfigurieren

Sie können die Write-Behind-Unterstützung über die ObjectGrid-XML-Deskriptordatei oder programmgesteuert über die Schnittstelle ″BackingMap″ aktivieren.

Verwenden Sie die ObjectGrid-XML-Deskriptordatei oder den programmgesteuerten Ansatz über die Schnittstelle ″BackingMap″, um die Write-Behind-Unterstützung zu aktivieren.

ObjectGrid-XML-Deskriptordatei

Wenn Sie ein ObjectGrid über eine ObjectGrid-XML-Deskriptordatei konfigurieren, wird der Write-Behind-Loader über das Attribut ″writeBehind″ im Tag ″backingMap″ aktiviert. Es folgt ein Beispiel:

```
<objectGrid name="library" >
     <backingMap name="book" writeBehind="T300;C900" pluginCollectionRef="bookPlugins"/>
```

Im vorherigen Beispiel wird die Write-Behind-Unterstützung für die BackingMap ″book″ mit dem Parameter ″T300;C900″ aktiviert. Das Attribut ″writeBehind″ gibt die maximale Aktualisierungszeit und/oder eine maximale Anzahl an Schlüsselaktualisierungen an. Der Parameter ″write-behind″ hat das folgende Format:

```
::= <defaults> | <Aktualisierungszeit> | <Anzahl der Schlüsselaktualisierungen> | <Aktualisierungszeit> ";"
<Anzahl der Schlüsselaktualisierungen>::= "T" <positive ganze Zahl>::= "C" <positive ganze Zahl>::= ""
```

- Attribut ″writeBehind″
- Aktualisierungszeit
- Anzahl der Schlüsselaktualisierungen
- defaults

Aktualisierungen im Loader finden statt, wenn eines der folgenden Ereignisse eintritt:

1. Die maximale Aktualisierungszeit in Sekunden seit der letzten Aktualisierung ist abgelaufen.
2. Die Anzahl aktualisierter Schlüssel in der Warteschlangen-Map hat die maximal zulässige Anzahl an Schlüsselaktualisierungen erreicht.

Diese Parameter sind lediglich Hinweise. Der echte Aktualisierungszähler und die echte Aktualisierungszeit liegen nah bei den Parametern. Es ist jedoch nicht garantiert, dass der echte Aktualisierungszähler und die echte Aktualisierungszeit den definierten Parametern entsprechen. Außerdem könnte die erste Write-Behind-Aktualisierung erst nach zwei Aktualisierungszeitintervallen stattfinden. Dies ist darauf zurückzuführen, dass ObjectGrid die Startzeit für die Aktualisierung zufällig wählt, so dass nicht alle Partitionen gleichzeitig auf die Datenbank zugreifen.

Im vorherigen Beispiel (T300;C900) schreibt der Loader die Daten 300 Sekunden nach der letzten Aktualisierung bzw. bei 900 zu aktualisierenden Schlüsseln in die

Datenbank. Die Standardaktualisierungszeit sind 300 Sekunden, und die Standardanzahl der Schlüsselaktualisierungen ist 1000.

Behandlung fehlgeschlagener Write-Behind-Aktualisierungen

Da die Transaktion von WebSphere eXtreme Scale vor dem Start der Back-End-Transaktion beendet wird, ist es möglich, dass eine erfolgreiche Transaktion berichtet wird, obwohl dies eigentlich nicht der Fall ist.

Wenn Sie beispielsweise versuchen, einen Eintrag in eine eXtreme-Scale-Transaktion einzufügen, die nicht in der BackingMap, aber im Back-End vorhanden ist führt dies zwar zu einem doppelten Schlüssel, aber die eXtreme-Scale-Transaktion ist trotzdem erfolgreich. Die Transaktion, in der der Write-Behind-Thread das Objekt in das Back-End einfügt, scheitert jedoch mit einer Ausnahme vom Typ "Schlüssel doppelt vorhanden".

Behandlung fehlgeschlagener Write-Behind-Aktualisierungen: Clientseite

Eine solche Aktualisierung und jede andere fehlgeschlagene Back-End-Aktualisierung ist eine fehlgeschlagene Write-Behind-Aktualisierung. Fehlgeschlagene Write-Behind-Aktualisierungen werden in einer Map für fehlgeschlagene Write-Behind-Aktualisierungen gespeichert. Diese Map dient als Ereigniswarteschlange für fehlgeschlagene Aktualisierungen. Der Schlüssel der Aktualisierung ist ein eindeutiges Integer-Objekt, und der Wert ist eine Instanz von FailedUpdateElement. Die fehlgeschlagene Write-behind-Aktualisierungs-Map ist mit einem Bereinigungsprogramm (Evictor) konfiguriert, der die Datensätze eine Stunde nach Einfügen entfernt. Deshalb gehen Datensätze der fehlgeschlagenen Aktualisierung verloren, wenn sie nicht innerhalb von einer Stunde abgerufen werden.

Mit der API "ObjectMap" können die Map-Einträge für fehlgeschlagene Write-Behind-Aktualisierung abgerufen werden. Die Map für fehlgeschlagene Write-Behind-Aktualisierungen hat den Namen "IBM_WB_FAILED_UPDATES_<Map-Name>". Die Präfixnamen für die einzelnen Write-Behind-System-Maps finden Sie in der Dokumentation zur API "WriteBehindLoaderConstants". Es folgt ein Beispiel.

Prozess fehlgeschlagen - Mustercode

```
ObjectMap failedMap = session.getMap(
    WriteBehindLoaderConstants.WRITE_BEHIND_FAILED_UPDATES_MAP_PREFIX + "Employee");
Object key = null;

session.begin();
while(key = failedMap.getNextKey(ObjectMap.QUEUE_TIMEOUT_NONE)) {
    FailedUpdateElement element = (FailedUpdateElement) failedMap.get(key);
    Throwable throwable = element.getThrowable();
    Object failedKey = element.getKey();
    Object failedValue = element.getAfterImage();
    failedMap.remove(key);
    // Schlüssel, Wert oder Ausnahme verarbeiten

}
session.commit();
```

Ein getNextKey-Aufruf arbeitet für jede eXtreme-Scale-Transaktion mit einer bestimmten Partition. In einer verteilten Umgebung müssen Sie zum Abrufen der Schlüssel aus allen Partitionen mehrere Transaktionen starten, wie im folgenden Beispiel gezeigt wird:

Schlüssel von allen Partitionen abrufen - Mustercode

```
ObjectMap failedMap = session.getMap(
    WriteBehindLoaderConstants.WRITE_BEHIND_FAILED_UPDATES_MAP_PREFIX + "Employee");
while (true) {
    session.begin();
    Object key = null;
    while(( key = failedMap.getNextKey(5000) )!= null ) {
        FailedUpdateElement element = (FailedUpdateElement) failedMap.get(key);
    Throwable throwable = element.getThrowable();
        Object failedKey = element.getKey();
        Object failedValue = element.getAfterImage();
```

```
        failedMap.remove(key);
        // Schlüssel, Wert oder Ausnahme verarbeiten

    }
    Session.commit();
}
```

Anmerkung: Die Map für fehlgeschlagene Aktualisierungen ist eine Möglichkeit, die Vitalität (den ordnungsgemäßen Betrieb) der Anwendung zu überwachen. Wenn ein System sehr viele Datensätze in der Map für fehlgeschlagene Aktualisierungen erzeugt, ist dies ein Hinweis darauf, dass die Anwendung bzw. Architektur überprüft bzw. so überarbeitet werden sollte, dass die Write-Behind-Unterstützung genutzt wird. Ab Version 6.1.0.5 können Sie das xsadmin-Script verwenden, um die Größe von Einträgen in der Map für fehlgeschlagene Aktualisierungen zu ermitteln.

Behandlung fehlgeschlagener Write-Behind-Aktualisierungen: Shard-Listener

Eine fehlgeschlagene Write-Behind-Transaktion sollte unbedingt erkannt und protokolliert werden. Jede Anwendung, die die Write-Behind-Technik verwendet, muss einen Watcher (Überwachungsprogramm) für die Behandlung fehlgeschlagener Write-Behind-Aktualisierungen implementieren. Auf diese Weise kann ein Speicherengpass verhindert werden, wenn Datensätze in der Map für fehlgeschlagene Aktualisierungen nicht entfernt werden, weil die Bereinigung durch die Anwendung erwartet wird.

Der folgende Code veranschaulicht, wie ein solcher Watcher oder Dumper integriert wird, der wie im folgenden Snippet der ObjectGrid-Deskriptor-XML hinzugefügt werden muss:

```
<objectGrid name="Grid">

        <bean id="ObjectGridEventListener" className="utils.WriteBehindDumper"/>
```

Wie Sie sehen, wurde die die Bean "ObjectGridEventListener" hinzugefügt. Diese Bean ist der zuvor erwähnte Write-Behind-Watcher. Der Watcher interagiert mit den Maps für alle primären Shards in einer JVM und sucht nach denen, in denen die Write-Behind-Technik aktiviert ist. Wenn er ein solches Shard findet, versucht er, bis zu 100 fehlgeschlagene Aktualisierungen zu protokollieren. Er überwacht ein primäres Shard so lange, bis das Shard in eine andere JVM verschoben wird. Alle Anwendungen, die die Write-Behind-Technik verwenden, *müssen* einen Watcher verwenden, der diesem gleicht. Andernfalls kann in der Java Virtual Machines ein Speicherengpass auftreten, weil die Fehler-Map nie bereinigt wird.

Weitere Informationen finden Sie im Abschnitt Mustercodes für eine Write-Behind-Dumper-Klasse.

Mustercode für eine Write-Behind-Dumper-Klasse

Dieser Musterquellcode veranschaulicht, wie Sie einen Watcher (Dumper) für die Behandlung fehlgeschlagener Write-Behind-Aktualisierungen schreiben.

```
//
// Dieses Musterprogramm wird ohne Wartung (auf "as-is"-Basis)
// bereitgestellt und kann vom Kunden (a) zu Schulungs- und Studienzwecken,
// (b) zum Entwickeln von Anwendungen für ein IBM WebSphere-Produkt zur
// internen Nutzung beim Kunden oder Weitergabe im Rahmen einer solchen
// Anwendung in kundeneigenen Produkten gebührenfrei genutzt, ausgeführt,
// kopiert und geändert werden."
//
//5724-J34 (C) COPYRIGHT International Business Machines Corp. 2009
//Alle Rechte vorbehalten * Lizenziertes Material - Eigentum von IBM
//
package utils;
```

```
import java.util.Collection;
import java.util.Iterator;
import java.util.concurrent.Callable;
import java.util.concurrent.ScheduledExecutorService;
import java.util.concurrent.ScheduledFuture;
import java.util.concurrent.ScheduledThreadPoolExecutor;
import java.util.concurrent.TimeUnit;
import java.util.logging.Logger;

import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.ObjectGridException;
import com.ibm.websphere.objectgrid.ObjectGridRuntimeException;
import com.ibm.websphere.objectgrid.ObjectMap;
import com.ibm.websphere.objectgrid.Session;
import com.ibm.websphere.objectgrid.UndefinedMapException;
import com.ibm.websphere.objectgrid.plugins.ObjectGridEventGroup;
import com.ibm.websphere.objectgrid.plugins.ObjectGridEventListener;
import com.ibm.websphere.objectgrid.writebehind.FailedUpdateElement;
import com.ibm.websphere.objectgrid.writebehind.WriteBehindLoaderConstants;

/**
 * Write behind expects transactions to the Loader to succeed. If a transaction for a key fails then
 * it inserts an entry in a Map called PREFIX + mapName. The application should be checking this
 * map for entries to dump out write behind transaction failures. The application is responsible for
 * analyzing and then removing these entries. These entries can be large as they include the key, before
 * and after images of the value and the exception itself. Exceptions can easily be 20k on their own.
 *
 * The class is registered with the grid and an instance is created per primary shard in a JVM. It creates a single thread
 * and that thread then checks each write behind error map for the shard, prints out the problem and then removes the
 * entry.
 *
 * This means there will be one thread per shard. If the shard is moved to another JVM then the deactivate method stops the
 * thread.
 * @author bnewport
 *
 */
public class WriteBehindDumper implements ObjectGridEventListener, ObjectGridEventGroup.ShardEvents, Callable<Boolean>
{
 static Logger logger = Logger.getLogger(WriteBehindDumper.class.getName());

 ObjectGrid grid;

 /**
  * Thread pool to handle table checkers. If the application has it's own pool
  * then change this to reuse the existing pool
  */
 static ScheduledExecutorService pool = new ScheduledThreadPoolExecutor(2); // two threads to dump records

 // the future for this shard
 ScheduledFuture<Boolean> future;

 // true if this shard is active
 volatile boolean isShardActive;

 /**
  * Normal time between checking Maps for write behind errors
  */
 final long BLOCKTIME_SECS = 20L;

 /**
  * An allocated session for this shard. No point in allocating them again and again
  */
 Session session;
 /**
  * When a primary shard is activated then schedule the checks to periodically check
  * the write behind error maps and print out any problems
  */
public void shardActivated(ObjectGrid grid) {
  try
  {
   this.grid = grid;
   session = grid.getSession();

   isShardActive = true;
   future = pool.schedule(this, BLOCKTIME_SECS, TimeUnit.SECONDS); // check every BLOCKTIME_SECS seconds initially
  }
  catch(ObjectGridException e)
  {
```

```
  throw new ObjectGridRuntimeException("Exception activating write dumper", e);
 }
}

/**
 * Mark shard as inactive and then cancel the checker
 */
public void shardDeactivate(ObjectGrid arg0)
{
 isShardActive = false;
 // if it's cancelled then cancel returns true
 if(future.cancel(false) == false)
 {
  // otherwise just block until the checker completes
  while(future.isDone() == false) // wait for the task to finish one way or the other
  {
   try
   {
    Thread.sleep(1000L); // check every second
   }
   catch(InterruptedException e)
   {
   }
  }
 }
}

/**
 * Simple test to see if the map has write behind enabled and if so then return
 * the name of the error map for it.
 * @param mapName The map to test
 * @return The name of the write behind error map if it exists otherwise null
 */
static public String getWriteBehindNameIfPossible(ObjectGrid grid, String mapName)
{
 BackingMap map = grid.getMap(mapName);
 if(map != null && map.getWriteBehind() != null)
 {
  return WriteBehindLoaderConstants.WRITE_BEHIND_FAILED_UPDATES_MAP_PREFIX + mapName;
 }
 else
        return null;
}

/**
 * This runs for each shard. It checks if each map has write behind enabled and if it does
  * then it prints out any write behind
 * transaction errors and then removes the record.
 */
public Boolean call()
{
 logger.fine("Called for " + grid.toString());
 try
 {
  // while the primary shard is present in this JVM
  // only user defined maps are returned here, no system maps like write behind maps are in
  // this list.
  Iterator<String> iter = grid.getListOfMapNames().iterator();
  boolean foundErrors = false;
  // iterate over all the current Maps
  while(iter.hasNext() && isShardActive)
  {
   String origName = iter.next();

   // if it's a write behind error map
   String name = getWriteBehindNameIfPossible(grid, origName);
   if(name != null)
   {
    // try to remove blocks of N errors at a time
    ObjectMap errorMap = null;
    try
    {
     errorMap = session.getMap(name);
    }
    catch(UndefinedMapException e)
    {
     // at startup, the error maps may not exist yet, patience...
     continue;
    }
```

```
    // try to dump out up to N records at once
    session.begin();
    for(int counter = 0; counter < 100; ++counter)
    {
     Integer seqKey = (Integer)errorMap.getNextKey(1L);
     if(seqKey != null)
     {
      foundErrors = true;
      FailedUpdateElement elem = (FailedUpdateElement)errorMap.get(seqKey);
      //
      // Your application should log the problem here
      logger.info("WriteBehindDumper ( " + origName + ") for key (" + elem.getKey() + ") Exception: " +
       elem.getThrowable().toString());
      //
      //
      errorMap.remove(seqKey);
     }
     else
      break;
    }
    session.commit();
   }
  } // do next map
  // loop faster if there are errors
  if(isShardActive)
  {
   // reschedule after one second if there were bad records
   // otherwise, wait 20 seconds.
   if(foundErrors)
    future = pool.schedule(this, 1L, TimeUnit.SECONDS);
   else
    future = pool.schedule(this, BLOCKTIME_SECS, TimeUnit.SECONDS);
  }
 }
 catch(ObjectGridException e)
 {
  logger.fine("Exception in WriteBehindDumper" + e.toString());
  e.printStackTrace();

  //don't leave a transaction on the session.
  if(session.isTransactionActive())
  {
   try { session.rollback(); } catch(Exception e2) {}
  }
 }
 return true;
}

public void destroy() {
 // TODO Auto-generated method stub

}

public void initialize(Session arg0) {
 // TODO Auto-generated method stub

}

public void transactionBegin(String arg0, boolean arg1) {
 // TODO Auto-generated method stub

}

public void transactionEnd(String arg0, boolean arg1, boolean arg2,
  Collection arg3) {
 // TODO Auto-generated method stub

}
}
```

Bereinigungsprogramme konfigurieren

Bereinigungsprogramme (Evictor) können über die ObjectGrid-XML-Deskriptordatei oder über das Programm konfiguriert werden.

Warum und wann dieser Vorgang ausgeführt wird

Informationen zur Konfiguration mit XML finden Sie im Abschnitt „ObjectGrid-XML-Deskriptordatei" auf Seite 157.

TTL-Bereinigungsprogramm aktivieren

WebSphere eXtreme Scale stellt einen Standardmechanismus für das Entfernen von Cacheeinträgen und ein Plug-in für das Erstellen angepasster Bereinigungsprogramme (Evictor) bereit. Ein Bereinigungsprogramm steuert die Zugehörigkeit von Einträgen in jeder BackingMap-Instanz. Das Standardbereinigungsprogramm verwendet eine Bereinigungsrichtlinie für jede BackingMap-Instanz, die auf der Lebensdauer (TTL, Time-to-Live) basiert. Wenn Sie ein Plug-in-fähiges Bereinigungsprogramm bereitstellen, verwendet dieses gewöhnlich eine Bereinigungsrichtlinie, die auf der Anzahl der Einträge und nicht auf der Lebensdauer basiert.

TTL-Bereinigungsprogramm programmgesteuert aktivieren

TTL-Bereinigungsprogramme werden BackingMap-Instanzen zugeordnet. Das folgende Code-Snippet veranschaulicht, wie die Schnittstelle "BackingMap" verwendet werden kann, um die erforderlichen Attribute zu setzen, so dass für jeden erstellten Eintrag die Verfallszeit gesetzt wird, die zehn Minuten nach der Erstellung des Eintrags abläuft:

TTL-Bereinigungsprogramm über das Programm aktivieren
```
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.TTLType;
ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid og = ogManager.createObjectGrid( "grid" );
BackingMap bm = og.defineMap( "myMap" );
bm.setTtlEvictorType( TTLType.CREATION_TIME );
bm.setTimeToLive( 600 );
```

Das Argument für die Methode "setTimeToLive" ist 600 und gibt die Lebensdauer in Sekunden an. Der vorherige Code muss vor dem Aufruf der Methode "initialize" in der ObjectGrid-Instanz aufgerufen werden. Die BackingMap-Attribute können nach der Initialisierung der ObjectGrid-Instanz nicht mehr geändert werden. Nach der Ausführung des Codes haben alle Einträge, die in die BackingMap "myMap" eingefügt werden, eine Verfallszeit. Nach Ablauf der Verfallszeit entfernt das TTL-Bereinigungsprogramm den Eintrag.

Wenn eine Anwendung erfordert, dass die Verfallszeit auf die letzte Zugriffszeit plus zehn Minuten gesetzt wird, muss nur eine einzige Zeile des vorherigen Codes geändert werden. Das Argument, das an die Methode "setTtlEvictorType" übergeben wird, muss von TTLType.CREATION_TIME in TTLType.LAST_ACCESS_TIME geändert werden. Mit diesem Wert wird die Verfallszeit mit der Formel "letzte Zugriffszeit plus zehn Minuten" berechnet. Bei der Erstellung eines Eintrags entspricht die letzte Zugriffszeit der Erstellungszeit.

Wenn das Argument TTLType.LAST_ACCESS_TIME verwendet wird, können die Schnittstellen "ObjectMap" und "JavaMap" verwendet werden, um den TTL-Wert in der BackingMap zu überschreiben. Dieser Mechanismus ermöglicht einer Anwendung, für jeden erstellten Eintrag einen anderen TTL-Wert zu verwenden. Angenommen, das vorherige Code-Snippet wurde verwendet, um das Attribut "ttlType" auf LAST_ACCESS_TIME zu setzen, und der TTL-Wert wurde in der BackingMap-Instanz auf zehn Minuten gesetzt. Anschließend kann eine Anwendung den TTL-Wert für jeden Eintrag ändern, indem sie den folgenden Code vor der Erstellung bzw. Änderung eines Eintrags ausführt:

```
import com.ibm.websphere.objectgrid.Session;
import com.ibm.websphere.objectgrid.ObjectMap;
Session session = og.getSession();
ObjectMap om = session.getMap( "myMap" );
int oldTimeToLive1 = om.setTimeToLive( 1800 );
om.insert("key1", "value1" );
int oldTimeToLive2 = om.setTimeToLive( 1200 );
om.insert("key2", "value2" );
```

Im vorherigen Code-Snippet hat der Eintrag mit dem Schlüssel ″key1″ aufgrund des Methodenaufrufs ″setTimeToLive(1800) in der ObjectMap eine Verfallszeit, die der Einfügezeit plus 30 Minuten entspricht. Die Variable ″oldTimeToLive1″ wird auf 600 gesetzt, weil der TTL-Wert aus der BackingMap als Standardwert verwendet wird, wenn die Methode ″setTimeToLive″ in der ObjectMap noch nicht aufgerufen wurde.

Der Eintrag mit dem Schlüssel ″key2″ hat aufgrund des Methodenaufrufs ″setTimeToLive(1200)″ in der ObjectMap eine Verfallszeit, die der Einfügezeit plus 20 Minuten entspricht. Die Variable ″oldTimeToLive2″ wird auf 1800 gesetzt, weil der TTL-Wert aus dem vorherigen Aufruf der Methode ″ObjectMap.setTimeToLive″ den TTL-Wert auf 1800 gesetzt hat.

Das vorherige Beispiel zeigt zwei Map-Einträge, die in die Map ″myMap″ für die Schlüsselwerte ″key1″ und ″key2″ eingefügt werden. Später möchte die Anwendung aus einem neuen Thread diese Map-Einträge mit neuen Map-Werten aktualisieren. Allerdings möchte die Anwendung für jeden Map-Eintrag die TTL-Werte beibehalten, die zur Einfügezeit verwendet wurden. Das folgende Beispiel veranschaulicht, wie die TTL-Werte durch Verwendung einer in der Schnittstelle ″ObjectMap″ definierten Konstanten beibehalten werden können:

```
Session session = og.getSession();
ObjectMap om = session.getMap( "myMap" );
om.setTimeToLive( ObjectMap.USE_DEFAULT );
session.begin();
om.update("key1", "updated value1" );
om.update("key2", "updated value2" );
om.insert("key3", "value3" );
session.commit();
```

Da der Sonderwert ObjectMap.USE_DEFAULT im Aufruf der Methode ″setTimeToLive″ verwendet wird, behält der Schlüssel ″key1″ seinen TTL-Wert von 1800 Sekunden und der Schlüssel ″key2″ seinen TTL-Wert von 1200 Sekunden bei, weil diese Werte verwendet wurden, als diese Map-Einträge durch die vorherige Transaktion eingefügt wurden.

Das vorherige Beispiel zeigt auch einen neuen Map-Eintrag für den eingefügten Schlüssel ″key3″. In diesem Fall gibt der Sonderwert USE_DEFAULT an, dass die TTL-Standardeinstellung für diese Map zu verwenden ist. Der Standardwert wird mit dem TTL-Attribut der BackingMap definiert. Weitere Informationen zur Definition des TTL-Attributs in der BackingMap-Instanz finden Sie in der Beschreibung der Attribute der Schnittstelle ″BackingMap″.

Informationen zur Methode ″setTimeToLive″ in den Schnittstellen ″ObjectMap″ und ″JavaMap″ finden Sie in der API-Dokumentation. In der Dokumentation werden Sie darauf hingewiesen, dass eine Ausnahme des Typs ″IllegalStateException″ ausgelöst wird, wenn die Methode ″BackingMap.getTtlEvictorType()″ einen anderen Wert als den TTLType.LAST_ACCESS_TIME-Wert zurückgibt. Die Schnittstellen ″ObjectMap″ und ″JavaMap″ können nur dann zum Überschreiben des TTL-Werts verwendet werden, wenn Sie den TTL-Bereinigungsprogrammtyp LAST_ACCESS_TIME verwenden. Diese Methode kann nicht zum Überschreiben des TTL-Werts verwendet werden, wenn der TTL-Bereinigungsprogrammtyp CREATION_TIME oder NONE verwendet wird.

XML-Konfiguration zum Aktivieren des TTL-Bereinigungsprogramms

Anstelle der Verwendung der Schnittstelle ″BackingMap″ für die programmgesteuerte Definition der BackingMap-Attribute für das TTL-Bereinigungsprogramm können Sie jede BackingMap-Instanz über eine XML-Datei konfigurieren. Der folgende Code veranschaulicht, wie Sie diese Attribute für drei verschiedene BackingMap-Maps definieren:

```
TTL-Bereinigungsprogramm über XML aktivieren
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
<objectGrids>
    <objectGrid name="grid1">
        <backingMap name="map1" ttlEvictorType="NONE" />
        <backingMap name="map2" ttlEvictorType="LAST_ACCESS_TIME" timeToLive="1800" />
        <backingMap name="map3" ttlEvictorType="CREATION_TIME" timeToLive="1200" />
    </objectgrid>
</objectGrids>
```

Das vorherige Beispiel zeigt, dass die BackingMap ″map1″ den TTL-Bereinigungsprogrammtyp NONE verwendet. Die BackingMap ″map2″ verwendet den TTL-Bereinigungsprogrammtyp LAST_ACCESS_TIME und hat einen TTL-Wert von 1800 Sekunden (oder 30 Minuten). Die BackingMap ″map3″ verwendet den TTL-Bereinigungsprogrammtyp CREATION_TIME und hat einen TTL-Wert von 1200 Sekunden (oder 20 Minuten).

Optionale Plug-in-Bereinigungsprogramme

Das TTL-Standardbereinigungsprogramm verwendet eine Bereinigungsrichtlinie, die auf Zeit basiert, und die Anzahl der Einträge in der BackingMap hat keine Auswirkung auf die Verfallszeit eines Eintrags. Sie können ein optionales Plug-in-Bereinigungsprogramm verwenden, um Einträge auf der Basis der Anzahl vorhandener Einträge an Stelle der Zeit zu entfernen.

Die folgenden optionalen Plug-in-Bereinigungsprogramme stellen einige gängige Algorithmen bereit, mit denen entschieden werden kann, welche Einträge entfernt werden sollen, wenn eine BackingMap ihr Größenlimit erreicht.

- Das Bereinigungsprogramm ″LRUEvictor″ verwendet einen LRU-Algorithmus (Least Recently Used), um zu entscheiden, welche Einträge entfernt werden sollen, wenn die BackingMap eine maximale Anzahl an Einträgen überschreitet.
- Das Bereinigungsprogramm ″LFUEvictor″ verwendet einen LFU-Algorithmus (Least Frequently Used), um zu entscheiden, welche Einträge entfernt werden sollen, wenn die BackingMap eine maximale Anzahl an Einträgen überschreitet.

Die BackingMap informiert ein Bereinigungsprogramm, wenn Einträge in einer Transaktion erstellt, geändert oder entfernt werden. Die BackingMap verfolgt diese Einträge und entscheidet, wann Einträge aus der BackingMap-Instanz entfernt werden müssen.

Eine BackingMap-Instanz hat keine Konfigurationsdaten für eine maximale Größe. Stattdessen werden Eigenschaften des Bereinigungsprogramms definiert, um das Verhalten des Bereinigungsprogramms zu steuern. Die Bereinigungsprogramme ″LRUEvictor″ und ″LFUEvictor″ haben beide eine Eigenschaft für die maximale Größe, die das Bereinigungsprogramm anweist, mit dem Entfernen von Einträgen zu beginnen, wenn die maximale Größe überschritten wird. Wie das TTL-Bereinigungsprogramm entfernen auch die Bereinigungsprogramme ″LRUEvictor″

und ″LFUEvictor″ einen Eintrag möglicherweise nicht sofort, wenn die maximale Anzahl an Einträgen erreicht ist, um die Auswirkungen auf die Leistung zu minimieren.

Wenn der LRU- bzw. LFU-Bereinigungsalgorithmus für eine bestimmte Anwendung nicht angemessen ist, können Sie eigene Bereinigungsprogramme schreiben, um eine eigene Bereinigungsstrategie zu erstellen.

Optionale Plug-in-Bereinigungsprogramme verwenden

Wenn Sie der BackingMap-Konfiguration ein Plug-in-Bereinigungsprogramm hinzufügen möchten, können Sie die programmgesteuerte Konfiguration oder die XML-Konfiguration verwenden.

Plug-in-Bereinigungsprogramm programmgesteuert integrieren

Da Bereinigungsprogramme (Evictor) BackingMaps zugeordnet werden, verwenden Sie die Schnittstelle ″BackingMap″, um das Plug-in-Bereinigungsprogramm anzugeben.Das folgende Code-Snippet ist ein Beispiel für die Angabe eines LRUEvictor-Bereinigungsprogramms für die BackingMap ″map1″ und eines LFUEvictor-Bereinigungsprogramms für die BackingMap-Instanz ″map2″:

Bereinigungsprogramm über das Programm einfügen

```
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor;
import com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor;
ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid og = ogManager.createObjectGrid( "grid" );
BackingMap bm = og.defineMap( "map1" );
LRUEvictor evictor = new LRUEvictor();
evictor.setMaxSize(1000);
evictor.setSleepTime( 15 );
evictor.setNumberOfLRUQueues( 53 );
bm.setEvictor(evictor);
bm = og.defineMap("map2");
LFUEvictor evictor2 = new LFUEvictor();
evictor2.setMaxSize(2000);
evictor2.setSleepTime( 15 );
evictor2.setNumberOfHeaps( 211 );
bm.setEvictor(evictor2);
```

Das vorherige Snippet zeigt, dass ein LRUEvictor-Bereinigungsprogramm für die BackingMap ″map1″ mit einer ungefähren maximalen Anzahl von 53.000 (53 * 1000) Einträgen verwendet wird. Das LFUEvictor-Bereinigungsprogramm wird für die BackingMap ″map2″ mit einer ungefähren maximalen Anzahl von 422.000 (211 * 2000) Einträgen verwendet. Das LRU- und das LFU-Bereinigungsprogramm haben jeweils eine Ruhezeiteigenschaft, die angibt, wie lange das Bereinigungsprogramm ruht, bis es aktiviert wird und prüft, ob Einträge entfernt werden müssen. Die Ruhezeit wird in Sekunden angegeben. Mit einem Wert von 15 Sekunden kann erreicht werden, dass der die Leistungseinbußen nicht zu hoch werden und die BackingMap nicht zu groß wird. Das Ziel ist, die größtmögliche Ruhezeit zu verwenden, aber gleichzeitig zu verhindern, dass die BackingMap zu groß wird.

Die Methode ″setNumberOfLRUQueues″ setzt die LRUEvictor-Eigenschaft, die angibt, wie viele LRU-Warteschlangen das Bereinigungsprogramm verwendet, um die LRU-Informationen zu verwalten. Es wird eine Sammlung von Warteschlangen verwendet, so dass nicht jeder Eintrag die LRU-Informationen in derselben Warteschlange hält. Dieser Ansatz kann die Leistung verbessern, indem die Anzahl der Map-Einträge, die in demselben Warteschlangenobjekt synchronisiert werden müssen, minimiert wird. Die Anzahl der Warteschlangen zu erhöhen ist eine geeignete Methode, die Auswirkungen zu minimieren, die das LRU-Bereinigungsprogramm auf die Leistung haben kann. Ein guter Ausgangspunkt für die Anzahl der Warte-

schlangen sind zehn Prozent der maximalen Eintragsanzahl. Die Verwendung einer Primzahl ist gewöhnlich besser als die Verwendung einer anderen Zahl. Die Methode ″setMaxSize″ gibt an, wie viele Einträge in jeder Warteschlange zulässig sind. Wenn eine Warteschlange die maximal zulässige Eintragsanzahl erreicht, werden die Einträge mit dem ältesten Verwendungsdatum in dieser Warteschlange entfernt, wenn das Bereinigungsprogramm das nächste Mal prüft, ob Einträge entfernt werden müssen.

Die Methode ″setNumberOfHeaps″ legt mit der LFUEvictor-Eigenschaft fest, wie viele binäre Objekte im Heap-Speicher der LFUEvictor für die Verwaltung der LFU-Informationen verwendet. Auch hier wird eine Sammlung verwendet, um die Leistung zu verbessern. Ein guter Ausgangspunkt sind zehn Prozent der maximalen Eintragsanzahl, und die Verwendung einer Primzahl ist gewöhnlich besser als die Verwendung einer anderen Zahl. Die Methode ″setMaxSize″ gibt an, wie viele Einträge in jedem Heap-Speicher zulässig sind. Wenn ein Heap-Speicher die maximal zulässige Eintragsanzahl erreicht, werden die Einträge, die am seltensten verwendet wurden, aus diesem Heap-Speicher entfernt, wenn das Bereinigungsprogramm das nächste Mal prüft, ob Einträge entfernt werden müssen.

XML-Konfiguration für die Integration eines Plug-in-Bereinigungsprogramms

Anstatt verschiedene APIs für die programmgesteuerte Integration eines Bereinigungsprogramms und die Definition seiner Eigenschaften zu verwenden, können Sie eine XML-Datei verwenden, um jede BackingMap einzeln zu konfigurieren, wie im folgenden Beispiel gezeigt wird:

Bereinigungsprogramm über XML einfügen

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
<objectGrids>
    <objectGrid name="grid">
        <backingMap name="map1" ttlEvictorType="NONE" pluginCollectionRef="LRU" />
        <backingMap name="map2" ttlEvictorType="NONE" pluginCollectionRef="LFU" />
    </objectGrid>
</objectGrids>
<backingMapPluginCollections>
    <backingMapPlugincollection id="LRU">
        <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
            <property name="maxSize" type="int" value="1000" description="set max size for each LRU queue" />
            <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
            <property name="numberOfLRUQueues" type="int" value="53" description="set number of LRU queues" />
        </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="LFU">
        <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor">
            <property name="maxSize" type="int" value="2000" description="set max size for each LFU heap" />
            <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
            <property name="numberOfHeaps" type="int" value="211" description="set number of LFU heaps" />
        </bean>
    </backingMapPluginCollection>
</backingMapPluginCollections>
</objectGridConfig>
```

Speicherbasierte Bereinigung

Alle integrierten Bereinigungsprogramme unterstützten die speicherbasierte Bereinigung, die in der Schnittstelle ″BackingMap″ aktiviert werden kann, indem das Attribut ″evictionTriggers″ der BackingMap auf MEMORY_USAGE_THRESHOLD gesetzt wird. Weitere Informationen zum Definieren des Attributs ″evictionTriggers″ in BackingMap finden Sie in den Referenzinformationen zur Schnittstelle ″BackingMap″ und zur Konfiguration von eXtreme Scale.

Die speicherbasierte Bereinigung basiert auf einem Schwellenwert für die Auslastung des Heap-Speichers. Wenn die speicherbasierte Bereinigung in der BackingMap aktiviert ist und die BackingMap ein integriertes Bereinigungsprogramm hat, wird der Schwellenwert für die Auslastung auf einen Standardprozentsatz des Gesamtspeichers gesetzt, wenn noch kein Schwellenwert definiert wurde.

Wenn Sie den Standardprozentsatz für den Auslastungsschwellenwert ändern möchten, setzen Sie die Eigenschaft ″memoryThresholdPercentage″ in den Container- und Servereigenschaftendateien für eXtreme-Scale-Serverprozesse. Zum Definieren des Schwellenwerts für die Zielauslastung in einem eXtreme-Scale-Clientprozess können Sie die MBean ″MemoryPoolMXBean″ verwenden. Weitere Informationen finden Sie auch in der Beschreibung der Datei ″containerServer-.props″ und im Abschnitt zum Starten von eXtreme-Scale-Serverprozessen.

Wenn die Speicherbelegung zur Laufzeit den Schwellenwert für die Zielauslastung überschreitet, beginnen speicherbasierte Bereinigungsprogramme mit dem Entfernen von Einträgen und versuchen, die Speicherbelegung unterhalb des Schwellenwerts für die Zielauslastung zu halten. Es gibt jedoch keine Garantien, dass die Bereinigung schnell genug ist, um potenzielle abnormale Speicherbedingungen zu verhindern, wenn die Laufzeitumgebung des Systems weiterhin so schnell Speicher belegt.

Plug-in-Bereinigungsprogramm integrieren

Da Bereinigungsprogramme (Evictor) BackingMaps zugeordnet werden, verwenden Sie die Schnittstelle ″BackingMap″, um das Plug-in-Bereinigungsprogramm anzugeben.

Plug-in-Bereinigungsprogramm programmgesteuert integrieren

Das folgende Code-Snippet ist ein Beispiel für die Angabe eines LRUEvictor-Bereinigungsprogramms für die BackingMap ″map1″ und eines LFUEvictor-Bereinigungsprogramms für die BackingMap ″map2″:

```
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.BackingMap;
import com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor;
import com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor;
ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid og = ogManager.createObjectGrid( "grid" );
BackingMap bm = og.defineMap( "map1" );
LRUEvictor evictor = new LRUEvictor();
evictor.setMaxSize(1000);
evictor.setSleepTime( 15 );
evictor.setNumberOfLRUQueues( 53 );
bm.setEvictor(evictor);
bm = og.defineMap("map2");
LFUEvictor evictor2 = new LFUEvictor();
evictor2.setMaxSize(2000);
evictor2.setSleepTime( 15 );
evictor2.setNumberOfHeaps( 211 );
bm.setEvictor(evictor2);
```

Das vorherige Snippet zeigt, dass ein LRUEvictor-Bereinigungsprogramm für die BackingMap ″map1″ mit einer ungefähren maximalen Anzahl von 53.000 (53 * 1000) Einträgen verwendet wird. Das LFUEvictor-Bereinigungsprogramm wird für die BackingMap ″map2″ mit einer ungefähren maximalen Anzahl von 422.000 (211

* 2000) Einträgen verwendet. Das LRU- und das LFU-Bereinigungsprogramm haben jeweils eine Ruhezeiteigenschaft, die angibt, wie lange das Bereinigungsprogramm ruht, bis es aktiviert wird und prüft, ob Einträge entfernt werden müssen. Die Ruhezeit wird in Sekunden angegeben. Mit einem Wert von 15 Sekunden kann erreicht werden, dass der die Leistungseinbußen nicht zu hoch werden und die BackingMap nicht zu groß wird. Das Ziel ist, die größtmögliche Ruhezeit zu verwenden, aber gleichzeitig zu verhindern, dass die BackingMap zu groß wird.

Die Methode ″setNumberOfLRUQueues″ setzt die LRUEvictor-Eigenschaft, die angibt, wie viele LRU-Warteschlangen das Bereinigungsprogramm verwendet, um die LRU-Informationen zu verwalten. Es wird eine Sammlung von Warteschlangen verwendet, so dass nicht jeder Eintrag die LRU-Informationen in derselben Warteschlange hält. Dieser Ansatz kann die Leistung verbessern, indem die Anzahl der Map-Einträge, die in demselben Warteschlangenobjekt synchronisiert werden müssen, minimiert wird. Die Anzahl der Warteschlangen zu erhöhen ist eine geeignete Methode, die Auswirkungen zu minimieren, die das LRU-Bereinigungsprogramm auf die Leistung haben kann. Ein guter Ausgangspunkt für die Anzahl der Warteschlangen sind zehn Prozent der maximalen Eintragsanzahl. Die Verwendung einer Primzahl ist gewöhnlich besser als die Verwendung einer anderen Zahl. Die Methode ″setMaxSize″ gibt an, wie viele Einträge in jeder Warteschlange zulässig sind. Wenn eine Warteschlange die maximal zulässige Eintragsanzahl erreicht, werden die Einträge mit dem ältesten Verwendungsdatum in dieser Warteschlange entfernt, wenn das Bereinigungsprogramm das nächste Mal prüft, ob Einträge entfernt werden müssen.

Die Methode ″setNumberOfHeaps″ legt mit der LFUEvictor-Eigenschaft fest, wie viele binäre Objekte im Heap-Speicher der LFUEvictor für die Verwaltung der LFU-Informationen verwendet. Auch hier wird eine Sammlung verwendet, um die Leistung zu verbessern. Ein guter Ausgangspunkt sind zehn Prozent der maximalen Eintragsanzahl, und die Verwendung einer Primzahl ist gewöhnlich besser als die Verwendung einer anderen Zahl. Die Methode ″setMaxSize″ gibt an, wie viele Einträge in jedem Heap-Speicher zulässig sind. Wenn ein Heap-Speicher die maximal zulässige Eintragsanzahl erreicht, werden die Einträge, die am seltensten verwendet wurden, aus diesem Heap-Speicher entfernt, wenn das Bereinigungsprogramm das nächste Mal prüft, ob Einträge entfernt werden müssen.

XML-Konfiguration für die Integration eines Plug-in-Bereinigungsprogramms

Anstatt verschiedene APIs für die programmgesteuerte Integration eines Bereinigungsprogramms und die Definition seiner Eigenschaften zu verwenden, können Sie eine XML-Datei verwenden, um jede BackingMap einzeln zu konfigurieren, wie im folgenden Beispiel gezeigt wird:

```
<?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>
```

HashIndex konfigurieren

Das HashIndex-Plug-in unterstützt die Schnittstellen ″MapIndex″ und ″MapRangeIndex″. Durch die ordnungsgemäße Definition und Verwendung von Indizes kann die Abfrageleistung erheblich verbessert werden.

Die folgenden Attribute können verwendet werden, um einen HashIndex über die ObjectGrid-XML-Implementierungsdeskriptordatei oder über das Programm zu konfigurieren:

- Name: Der Name des Index. Der Name muss für jede Map eindeutig sein. Der Name wird verwendet, um das Indexobjekt von der ObjectMap-Instanz für die BackingMap abzurufen.
- AttributeName: Die durch Kommas getrennten Namen der zu indexierenden Attribute. Bei Feldzugriffsindizes entsprechen die Attributnamen den Feldnamen. Bei Eigenschaftszugriffsindizes sind die Attributnamen die JavaBean-kompatiblen Eigenschaftsnamen. Wenn es nur einen einzigen Attributnamen gibt, ist der HashIndex ein Einzelattributindex , und wenn dieses Attribut eine Beziehung ist, zusätzlich auch ein Beziehungsindex. Werden mehrere Attributnamen angegeben, ist der HashIndex ein zusammengesetzter Index.
- FieldAccessAttribute: Wird für Maps verwendet, die keine Entitäts-Maps sind. Wenn diese Einstellung den Wert ″true″ hat, wird direkt über die Felder auf das Objekt zugegriffen. Wenn diese Einstellung nicht angegeben oder auf ″false″ gesetzt wird, wird die Getter-Methode des Attributs verwendet, um auf die Daten zuzugreifen.
- POJOKeyIndex: Wird für Maps verwendet, die keine Entitäts-Maps sind. Wenn diese Einstellung auf ″true″ gesetzt ist, überwacht der Index selbst das Objekt im Schlüsselteil der Map. Dies ist hilfreich, wenn der Schlüssel ein zusammengesetzter Schlüssel ist und in den Wert kein Schlüssel eingebettet ist. Wenn diese Einstellung nicht angegeben oder auf ″false″ gesetzt wird, überwacht der Index selbst das Objekt im Wertteil der Map.
- RangeIndex: Wenn diese Einstellung auf ″true″ gesetzt ist, ist die Bereichsindexierung aktiviert, und die Anwendung kann das abgerufene Indexobjekt in die Schnittstelle ″MapRangeIndex″ umsetzen. Wird die Eigenschaft ″RangeIndex″ mit dem Wert ″false″ konfiguriert, kann die Anwendung das abgerufene Indexobjekt nur in die Schnittstelle ″MapIndex″ umsetzen.

Gegenüberstellung von Einzelattribut-Hash-Index und zusammengesetztem HashIndex

Wenn die Eigenschaft ″AttributeName″ des HashIndex mehrere Attributnamen enthält, ist der HashIndex ein zusammengesetzter Index. Enthält die Eigenschaft nur einen einzigen Attributnamen, ist der HashIndex ein Einzelattributindex. Der Wert der Eigenschaft ″AttributeName″ eines zusammengesetzten HashIndex kann beispielsweise ″city,state,zipcode″ sein. Er enthält drei Attribute, die durch Kommas voneinander getrennt sind. Wenn der Wert der Eigenschaft ″AttributeName″ nur aus ″zipcode″ besteht, hat der HashIndex nur ein einziges Attribut, d. h., er ist ein Einzelattribut-HashIndex.

Ein zusammengesetzter HashIndex ist eine effiziente Methode für die Suche zwischengespeicherter Objekte, wenn die Suchkriterien viele Attribute umfassen. Ein solcher Index unterstützt jedoch keine Bereichsindexierung, und seine Eigenschaft ″RangeIndex″ muss auf ″false″ gesetzt werden.

Weitere Informationen finden Sie im Abschnitt zum zusammengesetzten HashIndex im *Administratorhandbuch*.

Beziehungs-HashIndex

Wenn das indexierte Attribut eines Einzelattribut-HashIndex eine Beziehung (mit einem oder mehreren Werten) ist, ist der HashIndex ein Beziehungs-HashIndex. Für einen Beziehungs-HashIndex muss die Eigenschaft auf ″false″ gesetzt werden.

Ein Beziehungs-Hashindex kann Abfragen beschleunigen, die zyklische Referenzen oder die Abfragefilter IS NULL, IS EMPTY, SIZE und MEMBER OF verwenden. Weitere Informationen finden Sie in den Abschnitten zur Abfrageoptimierung mit Indizes im *Programmierhandbuch*.

Bereichs-HashIndex

Wenn die Eigenschaft ″RangeIndex″ eines HashIndex auf ″true″ gesetzt ist, ist der HashIndex ein Bereichsindex und kann die Schnittstelle ″MapRangeIndex″ unterstützen. MapRangeIndex unterstützt Funktionen, mit denen Sie Daten über Bereichsfunktionen, wie z. B. größer als und/oder kleiner als, suchen können, wohingegen die Schnittstelle ″MapIndex″ nur Vergleichsfunktionen unterstützt. Für einen Einzelattributindex kann die Eigenschaft ″RangeIndex″ nur dann auf ″true″ gesetzt werden, wenn das indexierte Attribut den Typ ″Comparable″ (Vergleichbar) hat. Wird der Einzelattributindex von einer Abfrage verwendet, muss die Eigenschaft ″RangeIndex″ auf ″true″ gesetzt werden und das indexierte Attribut den Typ ″Comparable″ haben. Für einen Beziehungs-HashIndex und einen zusammengesetzte HashIndex muss die Eigenschaft ″RangeIndex″ auf ″false″ gesetzt werden.

Im Folgenden finden Sie eine Zusammenfassung zur Verwendung von Bereichsindizes.

Tabelle 7. Unterstützung für Bereichsindizes

Typ des HashIndex	**Unterstützung von Bereichsindizes**
Einzelattribut-HashIndex: indexierter Schlüssel bzw. indexiertes Attribut hat den Typ ″Comparable″	Ja
Einzelattribut-HashIndex: indexierter Schlüssel bzw. indexiertes Attribut hat nicht den Typ ″Comparable″	Nein
Zusammengesetzter HashIndex	Nein
Beziehungs-HashIndex	Nein

Abfrageoptimierung mit HashIndex

Durch die ordnungsgemäße Definition und Verwendung von Indizes kann die Abfrageleistung erheblich verbessert werden. eXtreme-Scale-Abfragen können integrierte HashIndex-Plug-ins verwenden, um die Leistung von Abfragen zu verbessern. Die Verwendung von Indizes kann die Abfrageleistung zwar erheblich verbessern kann, kann sich aber nachteilig auf die Leistung von Operationen für Transaktions-Maps auswirken.

Peer-to-Peer-Replikation mit JMS konfigurieren

Der JMS-basierte (Java Message Service) Peer-to-Peer-Replikationsmechanismus wird in der verteilten und in der lokalen Umgebung von WebSphere eXtreme Scale verwendet. JMS ist ein Kern-zu-Kern-Replikationsprozess und lässt die Übertragung von Datenaktualisierungen zwischen lokalen ObjectGrids und verteilten ObjectGrids zu. Mit diesem Mechanismus können Sie beispielsweise Datenaktualisierungen aus einem verteilten eXtreme-Scale-Grid in ein lokales eXtreme-Scale-Grid oder aus einem anderen Grid in einer anderen Systemdomäne verschieben.

Vorbereitungen

Der JMS-basierte Peer-to-Peer-Replikationsmechanismus basiert auf den integrierten JMS-basierten Schnittstellen ″ObjectGridEventListener″ und ″com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener″. Ausführliche Informationen zum Aktivieren des Peer-to-Peer-Replikationsmechanismus finden Sie im Abschnitt „JMS-Ereignis-Listener” auf Seite 146.

Weitere Informationen finden Sie im Abschnitt „Mechanismus für Clientinaktivierung aktivieren” auf Seite 81.

Im Folgenden sehen Sie ein XML-Konfigurationsbeispiel für die Aktivierung eines Peer-to-Peer-Replikationsmechanismus in einer eXtreme-Scale-Konfiguration:

Konfiguration der Peer-to-Peer-Replikation - XML-Beispiel

```
<bean id="ObjectGridEventListener"
className="com.ibm.websphere.objectgrid.plugins.JMSObjectGridEventListener">
 <property name="replicationRole" type="java.lang.String" value="DUAL_ROLES" description="" />
  <property name="replicationStrategy" type="java.lang.String" value="PUSH" description="" />
  <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String" value="defaultTCF" description="" />
  <property name="jms_topicJndiName" type="java.lang.String" value="defaultTopic" description="" />
  <property name="jms_topicName" type="java.lang.String" value="defaultTopic" description="" />
  <property name="jms_userid" type="java.lang.String" value="" description="" />
  <property name="jms_password" type="java.lang.String" value="" description="" />
  <property name="jndi_properties" type="java.lang.String"
 value="java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory;
 java.naming.provider.url=tcp://localhost:61616;connectionFactoryNames=defaultTCF;
 topic.defaultTopic=defaultTopic"
 description="jndi properties" />
     </bean>
```

Mechanismus für Clientinaktivierung aktivieren

In einer verteilten Umgebung von WebSphere eXtreme Scale gibt es auf der Clientseite standardmäßig einen nahen Cache, wenn die optimistische Sperrstrategie verwendet wird oder Sperren inaktiviert sind. Der nahe Cache enthält seine eigenen lokalen zwischengespeicherten Daten. Wenn ein Client von eXtreme Scale eine Aktualisierung festschreibt, wird diese Aktualisierung an den nahen Cache des Clients und an den Server gesendet. Andere Clients von eXtreme Scale erhalten die Aktualisierungsinformationen jedoch nicht und können daraufhin Daten haben, die nicht auf dem neuesten Stand sind.

Naher Cache

Anwendungen müssen sich dieses Problems potenziell veralteter Daten in Clients von eXtreme Scale bewusst sein. Sie können die integrierte JMS-basierte (Java Message Service) ObjectGridEventListener-Klasse ″com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener″ verwenden, um den Mechanismus für Clientinaktivierung in einer verteilten eXtreme-Scale-Umgebung, einem so genannten eXtreme-Scale-Grid, zu aktivieren.

Der Mechanismus für Clientinaktivierung ist die Lösung für das Problem veralteter Daten im nahen Cache des Clients in einer verteilten eXtreme-Scale-Umgebung. Diese Mechanismus stellt sicher, dass der nahe Cache des Clients mit Servern oder anderen Clients synchronisiert wird. Aber selbst mit diesem JMS-basierten Mechanismus für Clientinaktivierung wird der nahe Cache des Clients nicht sofort aktualisiert. Es tritt eine Verzögerung auf, wenn die Laufzeitumgebung von eXtreme Scale Aktualisierungen veröffentlicht.

Es sind zwei Modelle für den Mechanismus für Clientinaktivierung in einer verteilten eXtreme-Scale-Umgebung verfügbar:

- Client/Server-Modell: In diesem Modell haben alle Serverprozesse die Rolle ″Publisher″ (Bereitsteller), die alle Transaktionsänderungen an der vorgesehenen JMS-Destination veröffentlicht. Alle Clientprozesse haben die Rolle ″Receiver″ (Empfänger) und empfangen alle Transaktionsänderungen von der vorgesehenen JMS-Destination.
- Modell mit dem Client in zwei Rollen: In diesem Modell haben alle Serverprozesse nichts mit der JMS-Destination zu tun. Alle Clientprozesse übernehmen die Rollen ″Publisher″ und ″Receiver″ für die JMS-Destinations. Transaktionsänderungen, die auf Clientseite stattfinden, werden an der JMS-Destination veröffentlicht, und alle Clients empfangen diese Transaktionsänderungen.

Weitere Informationen zum Aktivieren des Mechanismus für die Clientinaktivierung finden Sie im Abschnitt „JMS-Ereignis-Listener” auf Seite 146.

Client/Server-Modell

In einem Client/Server-Modell haben die Server die Rolle ″JMS-Publisher″, und der Client hat die Rolle ″JMS-Receiver″.

XML-Beispiel für Client/Server-Modell

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="AgentObjectGrid">
      <bean id="ObjectGridEventListener"
        className="com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener">
        <property name="invalidationModel" type="java.lang.String" value="CLIENT_SERVER_MODEL" description="" />
        <property name="invalidationStrategy" type="java.lang.String" value="PUSH" description="" />
        <property name="mapsToPublish" type="java.lang.String" value="agent;profile;pessimisticMap" description="" />
        <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String" value="defaultTCF" description="" />
        <property name="jms_topicJndiName" type="java.lang.String" value="defaultTopic" description="" />
        <property name="jms_topicName" type="java.lang.String" value="defaultTopic" description="" />
        <property name="jms_userid" type="java.lang.String" value="" description="" />
        <property name="jms_password" type="java.lang.String" value="" description="" />
        <property name="jndi_properties" type="java.lang.String"
          value="java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory;java.naming.provider.url=
    tcp://localhost:61616;connectionFactoryNames=defaultTCF;topic.defaultTopic=defaultTopic"
          description="jndi properties" />
      </bean>

      <backingMap name="agent" readOnly="false" pluginCollectionRef="agent" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="28800" />
      <backingMap name="profile" readOnly="false" pluginCollectionRef="profile" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="pessimisticMap" readOnly="false" pluginCollectionRef="pessimisticMap" preloadMode="false"
        lockStrategy="PESSIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="excludedMap1" readOnly="false" pluginCollectionRef="excludedMap1" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="excludedMap2" readOnly="false" pluginCollectionRef="excludedMap2" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
    </objectGrid>
  </objectGrids>

  <backingMapPluginCollections>
    <backingMapPluginCollection id="agent">
```

```
      <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.AgentObjectTransformer" />
    </backingMapPluginCollection>
    <backingMapPluginCollection id="profile">
      <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.ProfileObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
        <property name="maxSize" type="int" value="2000" description="set max size for LRU evictor" />
        <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
        <property name="numberOfLRUQueues" type="int" value="50" description="set number of LRU queues" />
      </bean>
    </backingMapPluginCollection>

    <backingMapPluginCollection id="pessimisticMap" />
    <backingMapPluginCollection id="excludedMap1" />
    <backingMapPluginCollection id="excludedMap2" />
  </backingMapPluginCollections>

</objectGridConfig>
```

Modell mit dem Client in zwei Rollen

In diesem Modell übernimmt jeder Client sowohl die Rolle ″JMS-Publisher″ als auch die Rolle ″JMS-Receiver″. Der Client veröffentlicht alle festgeschriebenen Transaktionsänderungen an der vorgesehenen JMS-Destination und empfängt alle festgeschriebenen Transaktionsänderungen von anderen Clients. Der Server selbst hat in diesem Modell nichts mit JMS zu tun.

XML-Beispiel mit Zweirollenmodell

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="AgentObjectGrid">
      <bean id="ObjectGridEventListener"
        className="com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener">
        <property name="invalidationModel" type="java.lang.String" value="CLIENT_AS_DUAL_ROLES_MODEL" description="" />
        <property name="invalidationStrategy" type="java.lang.String" value="PUSH" description="" />
        <property name="mapsToPublish" type="java.lang.String" value="agent;profile;pessimisticMap" description="" />
        <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String" value="defaultTCF" description="" />
        <property name="jms_topicJndiName" type="java.lang.String" value="defaultTopic" description="" />
        <property name="jms_topicName" type="java.lang.String" value="defaultTopic" description="" />
        <property name="jms_userid" type="java.lang.String" value="" description="" />
        <property name="jms_password" type="java.lang.String" value="" description="" />
        <property name="jndi_properties" type="java.lang.String"
          value="java.naming.factory.initial=org.apache.activemq.jndi.ActiveMQInitialContextFactory;java.naming.provider.url=
     tcp://localhost:61616;connectionFactoryNames=defaultTCF;topic.defaultTopic=defaultTopic"
          description="jndi properties" />
      </bean>

      <backingMap name="agent" readOnly="false" pluginCollectionRef="agent" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="28800" />
      <backingMap name="profile" readOnly="false" pluginCollectionRef="profile" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="pessimisticMap" readOnly="false" pluginCollectionRef="pessimisticMap" preloadMode="false"
        lockStrategy="PESSIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="excludedMap1" readOnly="false" pluginCollectionRef="excludedMap1" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
      <backingMap name="excludedMap2" readOnly="false" pluginCollectionRef="excludedMap2" preloadMode="false"
        lockStrategy="OPTIMISTIC" copyMode="COPY_ON_READ_AND_COMMIT" ttlEvictorType="LAST_ACCESS_TIME"
        timeToLive="2700" />
    </objectGrid>
  </objectGrids>

  <backingMapPluginCollections>
    <backingMapPluginCollection id="agent">
      <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.AgentObjectTransformer" />
    </backingMapPluginCollection>
    <backingMapPluginCollection id="profile">
      <bean id="ObjectTransformer" className="com.ibm.ws.objectgrid.test.scenario.ProfileObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
        <property name="maxSize" type="int" value="2000" description="set max size for LRU evictor" />
        <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
        <property name="numberOfLRUQueues" type="int" value="50" description="set number of LRU queues" />
      </bean>
    </backingMapPluginCollection>

    <backingMapPluginCollection id="pessimisticMap" />
    <backingMapPluginCollection id="excludedMap1" />
    <backingMapPluginCollection id="excludedMap2" />
  </backingMapPluginCollections>

</objectGridConfig>
```

Änderungen an Peer-JVMs verteilen

Die Objekte ″LogSequence″ und ″LogElement″ kommunizieren Änderungen, die in einer eXtreme-Scale-Transaktion stattfinden über ein ObjectGridEventListener-Plug-in.

Weitere Informationen zur Verwendung von Java Message Service (JMS) für die Verteilung von Transaktionsänderungen finden Sie in den Informationen zur Verwendung von JMS für die Verteilung von Transaktionsänderungen im *Produktübersicht*.

Eine Voraussetzung ist, dass die ObjectGrid-Instanz vom ObjectGridManager zwischengespeichert wird. Weitere Einzelheiten finden Sie in den Informationen zu den createObjectGrid-Methoden. Die boolesche Eigenschaft ″cacheInstance″ muss auf ″true″ gesetzt werden.

Dieser Mechanismus muss nicht implementiert werden. Sie können einen integrierten Mechanismus für die Peer-to-Peer-Replikation verwenden, um diese Funktion zu nutzen. Weitere Informationen finden Sie in der Dokumentation zur Peer-to-Peer-Replikation mit JMS im *Administratorhandbuch*.

Eine Anwendung kann diese Objekte verwenden, um Änderungen, die in einem ObjectGrid vorgenommen werden, problemlos über einen Nachrichtentransport an die Peer-ObjectGrids in fernen Java Virtual Machines zu veröffentlichen und die Änderungen anschließend in dieser JVM anzuwenden. Die Klasse ″LogSequence-Transformer″ ist für die Aktivierung dieser Unterstützung kritisch. In diesem Abschnitt wird beschrieben, wie ein Listener mit einem JMS-Messaging-System für die Weitergabe der Nachrichten geschrieben wird. Zu diesem Zweck unterstützt eXtreme Scale die Übertragung von LogSequence-Objekten, die sich aus der Festschreibung einer eXtreme-Scale-Transaktion ergeben, über ein von IBM bereitgestelltes Plug-in an die Cluster-Member von WebSphere Application Server. Diese Funktion ist standardmäßig nicht aktiviert, kann aber konfiguriert werden. Wenn der Konsument oder Erzeuger jedoch kein WebSphere Application Server ist, kann die Verwendung eines externen JMS-Messaging-Systems erforderlich sein.

Mechanismus implementieren

Die Klasse ″LogSequenceTransformer″ und die APIs ″ObjectGridEventListener″, ″LogSequence″ und ″LogElement″ lassen die Verwendung jedes zuverlässigen Publish/Subscribe-Mechanismus für die Verteilung der Änderungen und die Filterung der zu verteilenden Maps zu. Die Snippets in diesem Abschnitt veranschaulichen, wie diese APIs mit JMS verwendet werden können, um ein Peer-to-Peer-ObjectGrid zu erstellen, das von Anwendungen gemeinsam genutzt wird, die sich auf verschiedenen Plattformen befinden, die einen gemeinsamen Nachrichtentransport verwenden.

Plug-in initialisieren

Das ObjectGrid ruft im Rahmen des Vertrags der Schnittstelle ″ObjectGridEventListener″ die Methode ″initialize″ des Plug-ins auf, wenn das ObjectGrid gestartet wird. Die Methode ″initialize″ muss seine JMS-Ressourcen, einschließlich Verbindungen, Sitzungen und Veröffentlichungskomponenten (so genannten Publishern), anfordern und den Thread für den JMS-Listener starten.

Die folgenden Beispiele zeigen die Methode ″initialize″:

Beispiel für die Methode initialize

```
public void initialize(Session session) {
        mySession = session;
        myGrid = session.getObjectGrid();
        try {
            if (mode == null) {
                throw new ObjectGridRuntimeException("No mode specified");
            }
            if (userid != null) {
                connection = topicConnectionFactory.createTopicConnection(userid, password);
            } else
                connection = topicConnectionFactory.createTopicConnection();

            // Die Verbindung muss gestartet werden, um Nachrichten zu empfangen.
            connection.start();

            // Die JMS_Sitzung ist nicht transaktionsorientiert (false).
            jmsSession = connection.createTopicSession(false, javax.jms.Session.AUTO_ACKNOWLEDGE);
            if (topic == null)
                if (topicName == null) {
                    throw new ObjectGridRuntimeException("Topic not specified");
                } else {
                    topic = jmsSession.createTopic(topicName);
                }
            publisher = jmsSession.createPublisher(topic);
            // Listener-Thread starten.
            listenerRunning = true;
            listenerThread = new Thread(this);
            listenerThread.start();
        } catch (Throwable e) {
            throw new ObjectGridRuntimeException("Cannot initialize", e);
        }
    }
```

Der Code zum Starten des Threads verwendet einen Java-SE-Thread (Java 2 Platform, Standard Edition). Wenn Sie einen Server von WebSphere Application Server Version 6.x oder WebSphere Application Server Version 5.x ausführen, verwenden Sie die API für asynchrone Beans, um diesen Dämon-Thread zu starten. Sie können auch die allgemeinen APIs verwenden. Im Folgenden sehen Sie ein Beispiel für ein Ersatz-Snippet, dass diese Aktion mit einem Arbeitsmanager veranschaulicht:

```
// Listener-Thread starten.
listenerRunning = true;
workManager.startWork(this, true);
```

Das Plug-in muss die Schnittstelle "Work" an Stelle der Schnittstelle "Runnable" implementieren. Außerdem müssen Sie eine Methode "release" hinzufügen, um die Variable "listenerRunning" auf "false" zu setzen. Das Plug-in muss mit einer WorkManager-Instanz in seinem Konstruktor bzw. bei Verwendung eines IoC-Containers (Inversion of Control) durch Injektion bereitgestellt werden.

Änderungen übertragen

Im Folgenden sehen Sie eine Beispielmethode "transactionEnd" für die Veröffentlichung der lokalen Änderungen, die in einem ObjectGrid vorgenommen werden. In diesem Beispiel wird JMS verwendet, aber Sie können jeden Nachrichtentransport verwenden, der zuverlässiges Publish/Subscribe-Messaging unterstützt.

Beispiel für die Methode transactionEnd

```
     // Diese Methode wird synchronisiert, um sicherzustellen,
    // dass die Nachrichten in der Reihenfolge veröffentlicht werden, in die
    // Transaktionen festgeschrieben werden. Falls die Veröffentlichung der Nachrichten
    // parallel gestartet wird, könnten die Empfänger die Map beschädigen,
    // da Löschanforderungen vor Einfügeanforderungen usw. ankommen könnten.
public synchronized void transactionEnd(String txid, boolean isWriteThroughEnabled, boolean committed,
Collection changes) {
        try {
            // Muss Write-through und festgeschrieben sein.
            if (isWriteThroughEnabled && committed) {
                // Folgen in eine Bytefeldgruppe (byte []) schreiben.
                ByteArrayOutputStream bos = new ByteArrayOutputStream();
                ObjectOutputStream oos = new ObjectOutputStream(bos);
                if (publishMaps.isEmpty()) {
                    // Gesamte Sammlung serialisieren
                    LogSequenceTransformer.serialize(changes, oos, this, mode);
                } else {
                    // LogSequence-Objekte auf der Basis des publishMaps-Inhalts filtern.
                    Collection publishChanges = new ArrayList();
                    Iterator iter = changes.iterator();
```

```
                    while (iter.hasNext()) {
                      LogSequence ls = (LogSequence) iter.next();
                      if (publishMaps.contains(ls.getMapName())) {
                          publishChanges.add(ls);
                      }
                    }
                    LogSequenceTransformer.serialize(publishChanges, oos, this, mode);
                }
                // Objektnachricht für die Änderungen erstellen.
                oos.flush();
                ObjectMessage om = jmsSession.createObjectMessage(bos.toByteArray());
                // Eigenschaften festlegen.
                om.setStringProperty(PROP_TX, txid);
                om.setStringProperty(PROP_GRIDNAME, myGrid.getName());
                // Übertragen.
                publisher.publish(om);
            }
        } catch (Throwable e) {
            throw new ObjectGridRuntimeException("Cannot push changes", e);
        }
    }
```

In dieser Methode werden verschiedene Instanzvariablen verwendet:

- Variable `jmsSession`: Eine JMS-Sitzung, die zum Veröffentlichen von Nachrichten verwendet wird. Sie wird bei der Initialisierung des Plug-ins erstellt.
- Variable `mode`: Der Verteilungsmodus.
- Variable `publishMaps`: Eine Gruppe, die die Namen der einzelnen Maps mit zu veröffentlichenden Änderungen enthält. Wenn die Variable leer ist, werden alle Maps veröffentlicht.
- Variable `publisher`: Ein TopicPublisher-Objekt, das während der Ausführung der Methode "initialize" des Plug-ins ausgeführt wird.

Aktualisierungsnachricht empfangen und anwenden

Im Folgenden sehen Sie eine Beispielmethode "run". Diese Methode wird in einer Schleife ausgeführt, bis die Anwendung die Schleife stoppt. In jeder Schleifeniteration wird versucht, eine JMS-Nachricht zu empfangen und auf das ObjectGrid anzuwenden.

Beispiel für die Methode run für JMS-Nachrichten

```
private synchronized boolean isListenerRunning() {
        return listenerRunning;
    }

       public void run () {
        try {
            System.out.println("Listener starting");
            // JMS-Sitzung für den Empfang der Nachrichten abrufen.
            // Nicht transaktionsorientiert.
            TopicSession myTopicSession;
            myTopicSession = connection.createTopicSession(false, javax.jms.Session.AUTO_ACKNOWLEDGE);
             // Subskribenten für das Topic abrufen. True gibt an, dass keine Nachrichten
            // empfangen werden, die über Publisher in dieser Verbindung übertragen
            // wurden. Sonst werden eigene Aktualisierungen empfangen.
            TopicSubscriber subscriber = myTopicSession.createSubscriber(topic,
       null, true);
            System.out.println("Listener started");
            while (isListenerRunning()) {
                ObjectMessage om = (ObjectMessage) subscriber.receive(2000);
                if (om != null) {
                    // Sitzung verwenden, die bei der Initialisierung übergeben wurde.
                    // Sehr wichtig, dass Write-through (Durchschreiben) hier nicht verwendet wird.
                    mySession.beginNoWriteThrough();
                    byte[] raw = (byte[]) om.getObject();
                    ByteArrayInputStream bis = new ByteArrayInputStream(raw);
                    ObjectInputStream ois = new ObjectInputStream(bis);
                    // LogSequence-Objekte dekomprimieren.
                    Collection collection = LogSequenceTransformer.inflate(ois,
           myGrid);
                    Iterator iter = collection.iterator();
                      while (iter.hasNext()) {
                        // Änderungen jeder Map entsprechend dem bei der
                        // Serialisierung des LogSequence-Objekts verwendeten
                        // Modus verarbeiten.
                        LogSequence seq = (LogSequence) iter.next();
                        mySession.processLogSequence(seq);
                    }
                    mySession.commit();
```

```
            } // Wenn eine Nachricht vorhanden ist
        } // while loop
        // Verbindung stoppen.
        connection.close();
    } catch (IOException e) {
        System.out.println("IO Exception: " + e);
    } catch (JMSException e) {
        System.out.println("JMS Exception: " + e);
    } catch (ObjectGridException e) {
        System.out.println("ObjectGrid exception: " + e);
        System.out.println("Caused by: " + e.getCause());
    } catch (Throwable e) {
        System.out.println("Exception : " + e);
    }
    System.out.println("Listener stopped");
}
```

JMS-Ereignis-Listener

Der JMSObjectGridEventListener unterstützt einen Mechanismus für die Inaktivierung des clientseitigen nahen Caches und einen Mechanismus für die Peer-to-Peer-Replikation. Er ist eine JMS-Implementierung (Java Message Service) der Schnittstelle "ObjectGridEventListener".

Der Mechanismus für die Clientinaktivierung kann in einer verteilten eXtreme-Scale-Umgebung verwendet werden, um sicherzustellen, dass die Daten im clientnahen Cache mit Servern oder anderen Clients synchronisiert werden. Ohne diese Funktion könnte der clientnahe Cache veraltete Daten enthalten. Aber selbst mit diesem JMS-basierten Mechanismus für Clientinaktivierung müssen Sie wegen der Verzögerung für die Laufzeitumgebung beim Veröffentlichen von Aktualisierungen das Zeitfenster für die Aktualisierung eines clientnahen Caches berücksichtigen.

Der Mechanismus für die Peer-to-Peer-Replikation kann in verteilten und lokalen eXtreme-Scale-Umgebungen verwendet werden. Er ist ein Kern-zu-Kern-Replikationsprozess und lässt die Übertragung von Datenaktualisierungen zwischen lokalen ObjectGrids und verteilten ObjectGrids zu. Mit diesem Mechanismus können Sie beispielsweise Datenaktualisierungen aus einem verteilten Grid in ein lokales ObjectGrid oder aus einem anderen Grid in einer anderen Systemdomäne verschieben.

Der JMSObjectGridEventListener erfordert, dass der Benutzer JMS- und JNDI-Informationen (Java Naming and Directory Interface) konfiguriert, damit die erforderlichen JMS-Ressourcen abgerufen werden. Außerdem müssen replikationsbezogene Eigenschaften ordnungsgemäß gesetzt werden. In einer JEE-Umgebung muss JNDI in Web- und EJB-Containern (Enterprise JavaBean) verfügbar sein. In diesem Fall ist die JNDI-Eigenschaft optional, sofern Sie keine externen JMS-Ressourcen abrufen möchten.

Dieser Ereignis-Listener hat Eigenschaften, die Sie über XML- oder programmgesteuerte Ansätze konfigurieren und nur für die Clientinaktivierung und/oder nur für die Peer-to-Peer-Replikation verwenden können. Die meisten Eigenschaften sind für die Anpassung des Verhaltens zum Erzielen der erforderlichen Funktionalität optional.

.

Weitere Informationen finden Sie in den Informationen zur API "JMSObjectGridEventListener".

JMSObjectGridEventListener-Plug-in erweitern

Das JMSObjectGridEventListener-Plug-in ermöglicht Peer-ObjectGrid-Instanzen, Aktualisierungen zu empfangen, wenn die Daten im Grind geändert oder entfernt wurden. Außerdem ermöglicht es die Benachrichtigung von Clients, wenn Einträge aktualisiert oder einem eXtreme-Scale-Grid entfernt werden. In diesem Abschnitt wird beschrieben, wie das JMSObjectGridEventListener-Plug-in erweitert werden kann, damit Anwendungen beim Empfang einer JMS-Nachricht benachrichtigt werden können. Dies ist äußerst hilfreich, wenn die Einstellung CLIENT_SERVER-_MODEL für die Clientinaktivierung verwendet wird.

Bei der Ausführung in der Empfängerrolle wird die überschriebene Methode ″JMS-ObjectGridEventListener.onMessage″ automatisch von der eXtreme-Scale-Laufzeitumgebung aufgerufen, wenn die JMSObjectGridEventListener-Instanz JMS-Nachrichtenaktualisierungen vom Grid empfängt. Diese Nachrichten schließen eine Sammlung von LogSequence-Objekten ein. Die LogSequence-Objekte werden an die Methode ″onMessage″ übergeben, und die Anwendung verwendet das LogSequence-Objekt, um die Cacheeinträge zu identifizieren, die eingefügt, gelöscht, aktualisiert oder ungültig gemacht wurden.

Zur Verwendung des onMessage-Erweiterungspunkts führen Anwendungen die folgenden Schritte aus:

1. Erstellen Sie eine neue Klasse, die die Klasse ″JMSObjectGridEventListener″ erweitert und die Methode ″onMessage″ überschreibt.
2. Konfigurieren Sie den erweiterten JMSObjectGridEventListener auf dieselbe Weise wie den ObjectGridEventListener für ObjectGrid.

Der erweiterte JMSObjectGridEventListener ist eine Unterklasse von JMSObjectGridEventListener und kann nur zwei Methoden überschreiben: die Methoden ″initialize″ (optional) und ″onMessage″. Wenn eine Unterklasse von JMSObjectGridEventListener ObjectGrid-Artefakte wie ObjectGrid oder Session in der Methode ″onMessage″ verwenden muss, kann sie diese Artefakte in der Methode ″initialize″ abrufen und als Instanzvariablen speichern. In der Methode ″onMessage″ können zwischengespeicherte ObjectGrid-Artefakte auch verwendet werden, um eine übergebene Sammlung von LogSequence-Objekten zu verarbeiten.

Anmerkung: Die überschriebene Methode ″initialize″ muss die Methode ″super.initialize″ aufrufen, um den übergeordneten JMSObjectGridEventListener entsprechend zu initialisieren.

Im Folgenden sehen Sie ein Beispiel für eine erweiterte JMSObjectGridEventListener-Klasse:

```
package com.ibm.websphere.samples.objectgrid.jms.price;

import java.util.*;
import com.ibm.websphere.objectgrid.*;
import com.ibm.websphere.objectgrid.plugins.LogElement;
import com.ibm.websphere.objectgrid.plugins.LogSequence;
import com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener;

public class ExtendedJMSObjectGridEventListener extends JMSObjectGridEventListener{
 protected static boolean debug = true;

    /**
     * Dieses Grid ist dem Listener zugeordnet.
     */
     ObjectGrid grid;

    /**
     * Die Sitzung, die dem Listener zugeordnet ist
     */
     Session session;
```

```
    String objectGridType;

    public List receivedLogSequenceList = new ArrayList();


 /* (keine Javadoc)
  * @see com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener
    #initialize(com.ibm.websphere.objectgrid.Session)
  */
public void initialize(Session session) {
  // Anmerkung: Wenn Sie ein ObjectGrid-Artefakt verwenden müssen, muss diese Klasse
  // ObjectGrid von der übergebenen Session-Instanz und die ObjectMap von der
  // Session-instanz für jede transaktionsorientierte ObjectGrid-Map-Operation abrufen.

  super.initialize(session);  // Die Methode initialize der Superklasse muss aufgerufen werden
  this.session = session; // Session-Instanz zwischenspeichern, falls sie
  //   zum Durchführen der Map-Operation benötigt wird
  this.grid = session.getObjectGrid(); // ObjectGrid abrufen, falls
  //   ObjectGrid-Informationen abgerufen werden müssen

  if (grid.getObjectGridType() == ObjectGrid.CLIENT)
   objectGridType = "CLIENT";
  else if (grid.getObjectGridType() == ObjectGrid.SERVER)
   objectGridType = "Server";

  if (debug)
   System.out.println("ExtendedJMSObjectGridEventListener[" +
     objectGridType + "].initialize() : grid = " + this.grid);
 }

 /* (keine Javadoc)
  * @see com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener
    #onMessage(java.util.Collection)
  */
 protected void onMessage(Collection logSequences) {
  System.out.println("ExtendedJMSObjectGridEventListener[" +
    objectGridType + "].onMessage(): ");

  Iterator iter = logSequences.iterator();

  while (iter.hasNext()) {
            LogSequence seq = (LogSequence) iter.next();

      StringBuffer buffer = new StringBuffer();
      String mapName = seq.getMapName();
      int size = seq.size();
      buffer.append("\nLogSequence[mapName=" + mapName + ", size=" + size + ",
     objectGridType=" + objectGridType
        + "]: ");

      Iterator logElementIter = seq.getAllChanges();
      for (int i = seq.size() - 1; i >= 0; --i) {
       LogElement le = (LogElement) logElementIter.next();
       buffer.append(le.getType() + " -> key=" + le.getCacheEntry().getKey() + ", ");
      }
      buffer.append("\n");

      receivedLogSequenceList.add(buffer.toString());

    if (debug) {
       System.out.println("ExtendedJMSObjectGridEventListener["
      + objectGridType + "].onMessage(): " + buffer.toString());
      }
  }
 }

 public String dumpReceivedLogSequenceList() {
  String result = "";
  int size = receivedLogSequenceList.size();
  result = result + "\nExtendedJMSObjectGridEventListener[" + objectGridType
    + "]: receivedLogSequenceList size = " + size + "\n";
for (int i = 0; i < size; i++) {
   result = result + receivedLogSequenceList.get(i) + "\n";
  }
  return result;
 }

 public String toString() {
  return "ExtendedJMSObjectGridEventListener["
    + objectGridType + " - " + this.grid + "]";
 }
}
```

Konfiguration

Die erweiterte JMSObjectGridEventListener-Klasse muss für den Mechanismus für die Clientinaktivierung und für den Mechanismus für die Peer-to-Peer-Replikation gleich konfiguriert werden. Im Folgenden sehen Sie ein Beispiel für die XML-Konfiguration:

```
<objectGrid name="PRICEGRID">
   <bean id="ObjectGridEventListener"
    className="com.ibm.websphere.samples.objectgrid.jms.
      price.ExtendedJMSObjectGridEventListener">
<property name="invalidationModel" type="java.lang.String" value="CLIENT_SERVER_MODEL" description="" />
    <property name="invalidationStrategy" type="java.lang.String"
     value="INVALIDATE" description="" />
    <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String"
     value="jms/TCF" description="" />
    <property name="jms_topicJndiName" type="java.lang.String"
     value="GRID.PRICEGRID" description="" />
    <property name="jms_topicName" type="java.lang.String"
     value="GRID.PRICEGRID" description="" />
<property name="jms_userid" type="java.lang.String" value="" description="" />
<property name="jms_password" type="java.lang.String" value="" description="" />
   </bean>
   <backingMap name="PRICE" pluginCollectionRef="PRICE"></backingMap>
       </objectGrid>
```

Anmerkung: Der Klassenname der Bean ″ObjectGridEventListener″ wird mit der erweiterten JMSObjectGridEventListener-Klasse mit denselben Eigenschaften wie die generische JMSObjectGridEventListener-Klasse konfiguriert.

Konfiguration mit XML-Dateien

WebSphere eXtreme Scale wird mit einer Sammlung von XML-Dateien konfiguriert. Jede XML-Datei hat einen bestimmten Zweck in der Konfiguration des Produkts.

Konfiguration der Implementierungstopologie

Verwenden Sie den XML-Implementierungsrichtliniendeskriptor, um Ihre Implementierungstopologie zu verwalten.

Die Implementierungsrichtlinie wird als XML-Datei codiert, die dem eXtreme-Scale-Container bereitgestellt wird. In der XML-Datei werden die folgenden Informationen angegeben:

- die Maps, die zu den einzelnen MapSets gehören,
- die Anzahl der Partitionen,
- die Anzahl synchroner und asynchroner Replikate.

Die Implementierungsrichtlinie steuert auch die folgenden Verteilungsverhalten:

- die Mindestanzahl aktiver Container, bevor die Verteilung stattfindet,
- die automatische Verteilung verloren gegangener Shards,
- die Verteilung jedes Shards einer einzelnen Partition an eine jeweils andere Maschine.

Weitere Informationen zur XML-Implementierungsrichtliniendatei finden Sie im Abschnitt „XML-Deskriptordatei für Implementierungsrichtlinie” auf Seite 151.

Endpunktinformationen werden in der dynamischen Umgebung nicht vorkonfiguriert. Es sind keine Servernamen oder Informationen zur physischen Topologie in der Implementierungsrichtlinie enthalten. Alle Shards in einem dynamischen Grid werden automatisch vom Katalogservice an die Container verteilt. Der Katalogservice verwendet die in der Implementierungsrichtlinie definierten Vorgaben, um die Verteilung der Shards automatisch zu verwalten. Mit dieser automati-

schen Shard-Verteilung lassen sich große Grids ohne großen Aufwand konfigurieren. Sie können der Umgebung bei Bedarf auch Server hinzufügen.

Anmerkung: In einer Umgebung mit WebSphere Application Server werden Stammgruppen mit mehr als 50 Membern nicht unterstützt.

Eine XML-Implementierungsrichtliniendatei wird während des Starts an einen eXtreme-Scale-Server übergeben. Eine Implementierungsrichtlinie muss zusammen mit einer ObjectGrid-XML-Datei verwendet werden. Die Implementierungsrichtlinie ist zum Starten eines Containers zwar nicht erforderlich, aber empfehlenswert. Die Implementierungsrichtlinie muss mit der verwendeten ObjectGrid-XML-Datei kompatibel sein. Für jedes objectgridDeployment-Element in der Implementierungsrichtlinie muss ein entsprechendes objectGrid-Element in der ObjectGrid-XML vorhanden sein. Die Maps im objectgridDeployment-Element müssen mit den backingMap-Elementen in der ObjectGrid-XML konsistent sein. Jedes backingMap-Element darf nur in einem einzigen mapSet-Element referenziert werden.

Im folgenden Beispiel soll die Datei `companyGridDpReplication.xml` mit der entsprechenden Datei `companyGrid.xml` gepaart werden:

companyGridDpReplication.xml

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

 <objectgridDeployment objectgridName="CompanyGrid">
  <mapSet name="mapSet1" numberOfPartitions="11"
   minSyncReplicas="1" maxSyncReplicas="1"
   maxAsyncReplicas="0" numInitialContainers="4">
   <map ref="Customer" />
   <map ref="Item" />
   <map ref="OrderLine" />
   <map ref="Order" />
  </mapSet>
 </objectgridDeployment>

</deploymentPolicy>
```

companyGrid.xml

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

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

</objectGridConfig>
```

Die Datei `companyGridDpReplication.xml` enthält ein MapSet, das in 11 Partitionen eingeteilt ist. Jede Partition muss genau ein synchrones Replikat haben. Die Anzahl der synchronen Replikate wird über die Attribute ″minSyncReplicas″ und ″maxSyncReplicas″ vorgegeben. Da das Attribut ″minSyncReplicas″ auf 1 gesetzt ist, muss jede Partition im MapSet mindestens ein verfügbares synchrones Replikat für die Verarbeitung von Schreiboperationen haben. Da ″maxSyncReplicas″ auf 1 gesetzt ist, darf jede Partition maximal ein einziges synchrones Replikat haben. Die Partitionen in diesem MapSet haben keine asynchronen Replikate.

Das Attribut ″numInitialContainers″ weist den Katalogservice an, die Verteilung zu verzögern, bis vier Container für die Unterstützung dieses ObjectGrids verfügbar sind. Das Attribut ″numInitialContainers″ wird ignoriert, sobald die angegebene Anzahl an Containern erreicht ist.

Die Datei `companyGridDpReplication.xml` demonstriert eine gängige Methode für die Konfiguration einer Implementierungsrichtlinie, aber eine Implementierungsrichtlinie bietet Ihnen noch mehr Steuerungselemente, mit denen Sie steuern können, wie und wann Ihr ObjectGrid in der Umgebung implementiert wird. Eine vollständige Beschreibung der Implementierungsrichtliniendeskriptordatei finden Sie im Abschnitt „XML-Deskriptordatei für Implementierungsrichtlinie".

Verteilte Topologie

Verteilte kohärente Caches bieten eine höhere Leistung, Verfügbarkeit und Skalierbarkeit, die Sie konfigurieren können.

WebSphere eXtreme Scale führt eine automatische gleichmäßige Verteilung der Server durch. Sie können weitere Server hinzufügen, ohne WebSphere eXtreme Scale erneut starten zu müssen. Das Hinzufügen zusätzlicher Server ohne Neustart von eXtreme Scale ermöglicht Ihnen die Verwendung einfacher Implementierungen und Implementierungen in Terabytegröße, in denen Tausende von Servern benötigt werden. Diese Implementierungstopologie ist flexibel. Mit dem Katalogservice können Sie Server hinzufügen und entfernen, um Ressourcen besser zu nutzen, ohne den gesamten Cache entfernen zu müssen. Zum Hinzufügen oder Entfernen eines Servers verwenden Sie den Befehl startOgServer mit der Option **-catalogServiceEndPoints**, um einen Containerserver zu starten, der mit dem Katalogservice kommuniziert. Alle Clients der verteilten Topologie kommunizieren mit dem Katalogservice über Internet Interoperability Object Protocol (IIOP). Alle Clients verwenden die Schnittstelle "ObjectGrid", um mit Servern zu kommunizieren.

Mit den dynamischen Konfigurationsfunktionen von WebSphere eXtreme Scale können dem System Ressourcen ohne großen Aufwand hinzugefügt werden. Container enthalten die Daten, und der Katalogservice funktioniert als Touchpoint für das Grid. Die Container sind für die Verwaltung der Daten zuständig. Der Katalogservice ist für die Weiterleitung der Anforderungen an die richtige Stelle bei der ersten Berührung, die Zuordnung von Speicherplatz in Hostcontainern und die Verwaltung von Vitalität (Status) und Verfügbarkeit des Gesamtsystems zuständig. Clients stellen eine Verbindung zu einem Katalogservice her, rufen eine Beschreibung der Containerservertopologie ab und kommunizieren dann bei Bedarf direkt mit jedem Server. Wenn sich die Servertopologie durch das Hinzufügen neuer Server oder durch den Ausfall anderer Server ändert, wird der Client automatisch an den entsprechenden Server weitergeleitet, der die Daten enthält.

Ein Katalogservice existiert gewöhnlich in einem eigenen Grid von Java Virtual Machines. Für die Verwaltung mehrerer Server kann ein einziger Katalogservice verwendet werden. Ein Container kann eigenständig in einer JVM gestartet oder zusammen mit anderen Containern für verschiedene Server in eine beliebige JVM geladen werden. Ein Client kann in jeder JVM ausgeführt werden und mit einem oder mehreren Servern kommunizieren. Ein Client kann auch in derselben JVM wie ein Container ausgeführt werden.

XML-Deskriptordatei für Implementierungsrichtlinie

Zum Konfigurieren einer Implementierungsrichtlinie verwenden Sie eine XML-Deskriptordatei für die Implementierungsrichtlinie.

In den folgenden Abschnitten finden Sie Definitionen der Elemente und Attribute in der XML-Implementierungsrichtliniendeskriptordatei. Im Abschnitt „Datei `deploymentPolicy.xsd`" auf Seite 155 finden Sie ein Beispiel für das XML-Implementierungsrichtlinienschema.

Element ″deploymentPolicy″

Das Element ″deploymentPolicy″ ist das Ausgangselement der XML-Implementierungsrichtliniendatei. Dieses Element konfiguriert den Namespace der Datei und die Schemaposition. Das Schema wird in der Datei `deploymentPolicy.xsd` definiert.

- Anzahl der Vorkommen: 1
- Untergeordnetes Element: objectgridDeployment

Element ″objectgridDeployment″

Das Element ″objectgridDeployment″ wird verwendet, um eine ObjectGrid-Instanz aus der ObjectGrid-XML-Datei zu referenzieren. Im Element ″objectgridDeployment″ können Sie Ihre Maps in MapSets unterteilen.

- Anzahl der Vorkommen: 1 oder mehr
- Untergeordnetes Element: mapSet

Attribute

objectgridName
Gibt den Namen des zu implementierenden ObjectGrids an. Dieses Attribut referenziert ein objectGrid-Element, das in der ObjectGrid-XML-Datei definiert ist. (Erforderlich)

```
<objectgridDeployment objectgridName="objectgridName"/>
```

Das Attribut ″objectgridName″ ist beispielsweise mit `CompanyGrid` in der Datei `companyGridDpReplication.xml` definiert. Das Attribut ″objectgridName″ referenziert das `CompanyGrid`, das in der Datei `companyGrid.xml` definiert ist. Weitere Informationen finden Sie im Abschnitt „ObjectGrid-XML-Deskriptordatei″ auf Seite 157.

Element ″mapSet″

Das Element ″mapSet″ wird verwendet, um Maps zu gruppieren. Die Maps in einem Element ″mapSet″ werden ähnlich partitioniert und repliziert. Jede Map muss zu genau einem MapSet gehören.

- Anzahl der Vorkommen: 1 oder mehr
- Untergeordnetes Element: map

Attribute

Name
Gibt den Namen des MapSets an. Dieses Attribut muss innerhalb des Elements ″objectgridDeployment″ eindeutig sein. (Erforderlich)

numberOfPartitions
Gibt die Anzahl der Partitionen für das MapSet an. Der Standardwert ist 1. Die Anzahl muss für die Anzahl der Container, in denen die Partitionen enthalten sind, angemessen sein. (Optional)

minSyncReplicas
Gibt die Mindestanzahl synchroner Replikate für jede Partition im MapSet an. Der Standardwert ist 1. Es werden erst Shards verteilt, wenn die Domäne die Mindestanzahl synchroner Replikate unterstützen kann. Für die Unterstützung des minSyncReplicas-Werts benötigen Sie einen Container mehr, als der minSyncReplicas-Wert vorgibt. Wenn die Anzahl synchroner Replikate unter den Wert von minSyncReplicas fällt, werden keine Schreiboperationen für diese Partitionen mehr zugelassen. (Optional)

maxSyncReplicas
Gibt die maximale Anzahl synchroner Replikate für jede Partition im MapSet an. Der Standardwert ist 0. Es werden keine weiteren synchronen Replikate für eine Partition verteilt, wenn eine Domäne diese Anzahl synchroner Replikate für diese bestimmte Partition erreicht. Das Hinzufügen von Containern, die dieses ObjectGrid unterstützen, kann zu einer höheren Anzahl synchroner Replikate führen, wenn der maxSyncReplicas-Wert noch nicht erreicht ist. (Optional)

maxAsyncReplicas
Gibt die maximale Anzahl asynchroner Replikate für jede Partition im MapSet an. Der Standardwert ist 0. Nachdem das primäre Shard und alle synchronen Replikate für eine Partition verteilt wurden, werden asynchrone Replikate verteilt, bis der maxAsyncReplicas-Wert erreicht ist. (Optional)

replicaReadEnabled
Wenn Sie dieses Attribut auf `true` setzen, werden Leseanforderungen an die primären Shards und die Replikate einer Partition verteilt. Wenn Sie das Attribut "replicaReadEnabled" auf `false` setzen, werden Leseanforderungen nur an das primäre Shard weitergeleitet. Der Standardwert ist "false". (Optional)

numInitialContainers
Gibt die Anzahl an eXtreme-Scale-Containern an, die erforderlich sind, bevor eine Erstverteilung für die Shards in diesem MapSet vorgenommen wird. Der Standardwert ist 1. Mit diesem Attribut kann CPU-Zeit und Netzbandbreite eingespart werden, wenn ein ObjectGrid online gebracht wird. (Optional)

Beim Starten eines eXtreme-Scale-Containers wird ein Ereignis an den Katalogservice gesendet. Sobald die Anzahl aktiver Container dem numInitialContainers-Wert für ein MapSet entspricht, verteilt der Katalogservice die Shards aus dem MapSet, sofern der minSyncReplicas-Wert ebenfalls erreicht ist. Wenn der numInitialContainers-Wert erreicht ist, kann jedes Ereignis des Typs "Containerstart" eine Neuverteilung noch nicht verteilter und bereits verteilter Shards auslösen. Wenn Sie in etwa wissen, wie viele Container für dieses MapSet gestartet werden, können Sie für "numInitialContainers" einen Wert festlegen, der dieser Zahl nahe kommt, um eine Neuverteilung nach jedem Containerstart zu vermeiden. Die Verteilung findet erst statt, wenn der numInitialContainers-Wert für das MapSet erreicht ist.

autoReplaceLostShards
Gibt an, ob verloren gegangene Shards an andere Container verteilt werden. Der Standardwert ist "true". Wenn ein Container gestoppt wird oder ausfällt, gehen die Shards, die im Container ausgeführt werden, verloren. Wenn ein primäres Shards verloren geht, wird eines seiner Replikate in das neue primäre Shard hochgestuft. Aufgrund dieser Hochstufung geht eines der Replikate verloren. In bestimmten Fällen ist es nicht wünschenswert, dass die verloren gegangenen Shards automatisch in verfügbaren Containern ersetzt werden. Wenn verloren gegangene Shards nicht automatisch ersetzt werden sollen, setzen Sie das Attribut "autoReplaceLostShards" auf `false`. Diese Einstellung hat keine Auswirkung auf die Hochstufungskette, sondern nur auf die Neuverteilung des letzten Shards in der Kette. (Optional)

developmentMode
Mit diesem Attribut können Sie die Verteilung eines Shards in Relation zu dessen Peer-Shards beeinflussen. Der Standardwert ist "true". Wenn Sie das Attribut "developmentMode" auf `false` setzen, werden nie zwei Shards derselben Partition an dieselbe Maschine verteilt. Denn Sie das Attribut "developmentMode" auf `true` setzen, können Shards derselben Partition an dieselbe

Maschine verteilt werden. Für beide Fälle gilt jedoch, dass zwei Shards derselben Partition nie an denselben Container verteilt werden. (Optional)

placementStrategy
Es gibt zwei Verteilungsstrategien. Die Standardstrategie ist `FIXED_PARTITION`. Bei dieser Strategie entspricht die Anzahl primärer Shards, die auf verfügbare Container verteilt wird, der Summe aus Anzahl definierter Partionen und Anzahl der Replikate. Die alternativ Strategie ist `PER_CONTAINER`. Bei dieser Strategie entspricht die Anzahl primärer Shards, die auf jeden Container verteilt wird, der Anzahl der definierten Partitionen mit einer entsprechenden Anzahl an Replikaten, die auf andere Container verteilt werden. (Optional)

```
<mapSet
(1)   name="mapSetName"
(2)   numberOfPartitions="numberOfPartitions"
(3)   minSyncReplicas="minimumNumber"
(4)    maxSyncReplicas="maximumNumber"
(5)   maxAsyncReplicas="maximumNumber"
(6)   replicaReadEnable="true" | "false"
(7)   numInitialContainers="numberOfInitialContainersBeforePlacement"
(8)   autoReplaceLostShards="true" | "false"
(9)   developmentMode="true" | "false"
(10)   placementStrategy="FIXED_PARTITION"|"PER_CONTAINER"
/>
```

Im folgenden Beispiel wird das Element ″mapSet″ verwendet, um eine Implementierungsrichtlinie zu konfigurieren. Der Wert wird auf `mapSet1` gesetzt und in 10 Partitionen eingeteilt. Jede dieser Partitionen muss mindestens ein verfügbares synchrones Replikat und darf nicht mehr als zwei synchrone Replikate haben. Außerdem hat jede Partition ein asynchrones Replikat, wenn dies von der Umgebung unterstützt werden kann. Alle synchronen Replikate werden vor den asynchronen Replikaten verteilt. Der Katalogservice versucht auch erst dann, die Shards für mapSet1 zu verteilen, wenn die Domäne den minSyncReplicas-Wert unterstützt. Für die Unterstützung des minSyncReplicas-Werts sind mindestens zwei Container erforderlich: einer für das primäre Shard und zwei für das synchrone Replikat:

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

 <objectgridDeployment objectgridName="CompanyGrid">
  <mapSet name="mapSet1" numberOfPartitions="10"
   minSyncReplicas="1" maxSyncReplicas="2" maxAsyncReplicas="1"
    numInitialContainers="10" autoReplaceLostShards="true"
   developmentMode="false" replicaReadEnabled="true">
   <map ref="Customer"/>
   <map ref="Item"/>
   <map ref="OrderLine"/>
   <map ref="Order"/>
  </mapSet>
 </objectgridDeployment>

</deploymentPolicy>
```

Obwohl nur zwei Container erforderlich sind, um die Replikationseinstellungen zu erfüllen, erfordert das Attribut ″numInitialContainers″ 10 verfügbare Container, bevor der Katalogservice versucht, Shards aus diesem MapSet zu verteilen. Sobald die Domäne 10 Container hat, die das ObjectGrid ″CompanyGrid″ unterstützen können, werden alle Shards in mapSet1 verteilt.

Da das Attribut ″autoReplaceLostShards″ auf ″true″ gesetzt ist, werden alle Shards in diesem MapSet, die aufgrund eines Containerausfalls verloren gegangen sind, automatisch in anderen Containern ersetzt, sofern ein Container verfügbar ist, der das verloren gegangene Shard aufnehmen kann. Shards derselben Partition können für mapSet1 nicht an dieselbe Maschine verteilt werden, weil das Attribut ″developmentMode″ auf `false` gesetzt ist. Reine Leseanforderungen werden auf das pri-

märe Shard und dessen Replikate jeder Partition verteilt, weil ″replicaReadEnabled″ den Wert `true` hat.

Element ″map″

Jedes Element ″map″ in einem Element ″mapSet″ referenziert eines der backingMap-Elemente, die in der ObjectGrid-XML-Datei definiert sind. Jede Map in einer verteilten Instanz von eXtreme Scale muss zu genau einem Element ″mapSet″ gehören. Weitere Informationen zur ObjectGrid-XML-Datei finden Sie im Abschnitt „Datei `objectGrid.xsd`″ auf Seite 174.

- Anzahl der Vorkommen: 1 oder mehr
- Untergeordnetes Element: Ohne

Attribute

ref

Ist eine Referenz auf ein backingMap-Element in der ObjectGrid-XML-Datei. Jede Map in einem MapSet muss auf eine BackingMap aus der ObjectGrid-XML-Datei verweisen. Der Wert des Attributs ″ref″ muss dem Attribut ″name″ eines der backingMap-Elemente in der ObjectGrid-XML-Datei entsprechen. (Erforderlich)

```
<map
(1)   ref="backingMapReference"
/>
```

Die Datei `companyGridDpMapSetAttr.xml` verwendet das Attribut ″ref″ in der Map, um jedes der backingMap-Elemente in der Datei `companyGrid.xml` zu referenzieren.

Informationen zur Vermeidung von XML-Konfigurationskonflikten finden Sie im Abschnitt „Fehler in der XML-Konfiguration beheben″ auf Seite 86.

Datei `deploymentPolicy.xsd`:

Verwenden Sie das XML-Implementierungsrichtlinienschema, um eine XML-Implementierungsrichtliniendatei zu erstellen.

Im Abschnitt „XML-Deskriptordatei für Implementierungsrichtlinie″ auf Seite 151 finden Sie Beschreibungen der Elemente und Attribute, die in der Datei `deploymentPolicy.xsd` definiert sind.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:dp="http://www.ibm.com/ws/objectgrid/deploymentPolicy"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNameSpace="http://ibm.com/ws/objectgrid/deploymentPolicy"
 elementFormDefault="qualified">

 <xsd:element name="deploymentPolicy">
  <xsd:complexType>
   <xsd:choice>
    <xsd:element name="objectgridDeployment"
     type="dp:objectgridDeployment" minOccurs="1"
     maxOccurs="unbounded">
     <xsd:unique name="mapSetNameUnique">
      <xsd:selector xpath="dp:mapset" />
      <xsd:field xpath="@name" />
     </xsd:unique>
    </xsd:element>
   </xsd:choice>
  </xsd:complexType>
 </xsd:element>

 <xsd:complexType name="objectgridDeployment">
  <xsd:sequence>
   <xsd:element name="mapSet" type="dp:mapSet"
    maxOccurs="unbounded" minOccurs="1">
    <xsd:unique name="mapNameUnique">
     <xsd:selector xpath="dp:map" />
     <xsd:field xpath="@ref" />
    </xsd:unique>
```

```
  </xsd:element>
 </xsd:sequence>
 <xsd:attribute name="objectgridName" type="xsd:string"
  use="required" />
</xsd:complexType>

<xsd:complexType name="mapSet">
 <xsd:sequence>
  <xsd:element name="map" type="dp:map" maxOccurs="unbounded"
   minOccurs="1" />
  <xsd:element name="zoneMetadata" type="dp:zoneMetadata"
   maxOccurs="1" minOccurs="0">

   <xsd:key name="zoneRuleName">
    <xsd:selector xpath="dp:zoneRule" />
    <xsd:field xpath="@name" />
   </xsd:key>

   <xsd:keyref name="zoneRuleRef"
    refer="dp:zoneRuleName">
    <xsd:selector xpath="dp:shardMapping" />
    <xsd:field xpath="@zoneRuleRef" />
   </xsd:keyref>

  </xsd:element>
 </xsd:sequence>
 <xsd:attribute name=="name" type="xsd:string" use="required" />
 <xsd:attribute name="numberOfPartitions" type="xsd:int"
  use="optional" />
 <xsd:attribute name="minSyncReplicas" type="xsd:int"
  use="optional" />
 <xsd:attribute name="maxSyncReplicas" type="xsd:int"
  use="optional" />
 <xsd:attribute name="maxAsyncReplicas" type="xsd:int"
  use="optional" />
 <xsd:attribute name="replicaReadEnabled" type="xsd:boolean"
  use="optional" />
 <xsd:attribute name="numInitialContainers" type="xsd:int"
  use="optional" />
 <xsd:attribute name="autoReplaceLostShards" type="xsd:boolean"
  use="optional" />
 <xsd:attribute name="developmentMode" type="xsd:boolean"
  use="optional" />
 <xsd:attribute name="placementStrategy"
  type="dp:placementStrategy" use="optional" />
</xsd:complexType>

<xsd:simpleType name="placementStrategy">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="FIXED_PARTITIONS" />
  <xsd:enumeration value="PER_CONTAINER" />
 </xsd:restriction>
</xsd:simpleType>


<xsd:complexType name="map">
 <xsd:attribute name="ref" use="required" />
</xsd:complexType>

<xsd:complexType name="zoneMetadata">
 <xsd:sequence>
  <xsd:element name="shardMapping" type="dp:shardMapping"
   maxOccurs="unbounded" minOccurs="1" />
  <xsd:element name="zoneRule" type="dp:zoneRule"
   maxOccurs="unbounded" minOccurs="1">

  </xsd:element>

 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="shardMapping">
 <xsd:attribute name="shard" use="required">
  <xsd:simpleType>
   <xsd:restriction base="xsd:string">
    <xsd:enumeration value="P"></xsd:enumeration>
    <xsd:enumeration value="S"></xsd:enumeration>
    <xsd:enumeration value="A"></xsd:enumeration>
   </xsd:restriction>
  </xsd:simpleType>
 </xsd:attribute>
 <xsd:attribute name="zoneRuleRef" type="xsd:string"
  use="required" />
</xsd:complexType>

<xsd:complexType name="zoneRule">
 <xsd:sequence>
  <xsd:element name="zone" type="dp:zone"
   maxOccurs="unbounded" minOccurs="1" />
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required" />
```

```
  <xsd:attribute name="exclusivePlacement" type="xsd:boolean" />
   use="optional" />
 </xsd:complexType>

 <xsd:complexType name="zone">
  <xsd:attribute name="name" type="xsd:string" use="required" />
 <xsd:complextType>

</xsd:schema>
```

ObjectGrid-XML-Deskriptordatei

Verwenden Sie zum Konfigurieren von WebSphere eXtreme Scale eine ObjectGrid-XML-Deskriptordatei und die API "ObjectGrid".

In den folgenden Abschnitten werden XML-Musterdateien bereitgestellt, um verschiedene Konfigurationen zu veranschaulichen. Jedes Element und Attribut der XML-Datei wird definiert. Verwenden Sie das ObjectGrid-XML-Deskriptorschema, um die XML-Deskriptordatei zu erstellen. Ein Beispiel für die ObjectGrid-XML-Deskriptordatei finden Sie im Abschnitt „Datei objectGrid.xsd" auf Seite 174.

Es wird eine geänderte Version der ursprünglichen Datei companyGrid.xml verwendet. Die folgende Datei companyGridSingleMap.xml ist der Datei companyGrid.xml ähnlich. Die Datei companyGridSingleMap.xml hat eine einzige Map, und die Datei companyGrid.xml hat vier Maps. Die Elemente und Attribute der Datei sind im Anschluss an das Beispiel detailliert beschrieben.

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

  <objectGrids>
<objectGrid name="CompanyGrid">
      <backingMap name="Customer"/>
    </objectGrid>
  </objectGrids>
</objectGridConfig>
```

Element "objectGridConfig"

Das Element "objectGridConfig" ist das Ausgangselement der XML-Datei. Schreiben Sie dieses Element in Ihrem eXtreme-Scale-XML-Dokument, wie im vorherigen Beispiel gezeigt. Dieses Element konfiguriert den Namespace der Datei und die Schemaposition. Das Schema ist in der Datei objectGrid.xsd definiert.

- Anzahl der Vorkommen: 1
- Untergeordnete Elemente: objectGrids und backingMapPluginCollections

Element "objectGrids"

Das Element "objectGrids" ist ein Container für alle objectGrid-Elemente. In der Datei companyGridSingleMap.xml enthält das Element "objectGrids" ein einziges ObjectGrid mit dem Namen "CompanyGrid".

- Anzahl der Vorkommen: 1 oder mehr
- Untergeordnetes Element: objectGrid

Element "objectGrid"

Verwenden Sie das Element "objectGrid", um ein ObjectGrid zu definieren. Jedes der Attribute im Element "objectGrid" entspricht einer Methode in der Schnittstelle "ObjectGrid".

- Anzahl der Vorkommen: 1 bis viele

- Untergeordnete Elemente: bean, backingMap, querySchema und streamQuerySet

Attribute

name
Gibt den Namen an, der dem ObjectGrid zugeordnet wird. Die XML-Validierung schlägt fehl, wenn dieses Attribut fehlt. (Erforderlich)

securityEnabled
Wenn Sie dieses Attribut auf `true` setzen, wird die Sicherheit auf ObjectGrid-Ebene aktiviert, woraufhin die Zugriffberechtigungen für die Daten in der Map aktiviert werden. Der Standardwert ist `true`. (Optional)

authorizationMechanism
Legt den Berechtigungsmechanismus für das Element fest. Sie können das Attribut auf einen der folgenden Werte setzen: `AUTHORIZATION_MECHANISM_JAAS` oder `AUTHORIZATION_MECHANISM_CUSTOM`. Der Standardwert ist `AUTHORIZATION_MECHANISM_JAAS`. Setzen Sie das Attribut auf `AUTHORIZATION_MECHANISM_CUSTOM`, wenn Sie ein angepasstes MapAuthorization-Plug-in verwenden. Sie müssen das Attribut "securityEnabled" auf `true` setzen, damit das Attribut "authorizationMechanism" wirksam wird. (Optional)

permissionCheckPeriod
Gibt einen ganzzahligen Wert in Sekunden an, der angibt, wie oft die Berechtigung geprüft wird, die verwendet wird, um einen Clientzugriff zuzulassen. Der Standardwert ist `0`. Wenn Sie das Attribut auf `0` setzen, wird der Berechtigungsmechanismus (Java Authentication and Authorization Service (JAAS) oder angepasste Berechtigung) bei jedem Aufruf der Methoden "get", "put", "update", "remove" und "evict" aufgefordert, zu prüfen, ob das aktuelle Subject-Objekt berechtigt ist. Ein Wert größer als 0 gibt die Anzahl an Sekunden an, für die eine Gruppe von Berechtigungen zwischengespeichert wird, bevor sie zur Aktualisierung an den Berechtigungsmechanismus zurückgegeben wird. Sie müssen das Attribut "securityEnabled" auf `true` setzen, damit das Attribut "permissionCheckPeriod" wirksam wird. (Optional)

txTimeout
Gibt die Zeit in Sekunden an, die einer Transaktion für die Ausführung zugestanden wird. Wenn eine Transaktion nicht innerhalb dieser Zeit abgeschlossen wird, wird sie für Rollback markiert, und es wird eine Ausnahme des Typs "TransactionTimeoutException" ausgelöst. Wenn Sie das Attribut auf den Wert `0` setzen, existiert kein Zeitlimit für die Transaktionsausführung. (Optional)

entityMetadataXMLFile
Gibt den relativen Pfad zur XML-Deskriptordatei der Entität an. Der Pfad ist relativ zur Position der Objectgrid-Deskriptordatei. Verwenden Sie dieses Attribut, um ein Entitätsschema über eine XML-Datei zu definieren. Entitäten müssen vor dem Starten von eXtreme Scale definiert werden, so dass jede Entität an eine BackingMap gebunden werden kann. (Optional)

```
<objectGrid
(1) name="objectGridName"
(2) securityEnabled="true" | "false"
(3) authorizationMechanism="AUTHORIZATION_MECHANISM_JASS" | "AUTHORIZATION_MECHANISM_CUSTOM"
(4)  permissionCheckPeriod="permission_check_period"
(5) txTimeout="seconds"
(6) entityMetadataXMLFile="URL"
/>
```

Im folgenden Beispiel demonstriert die Datei `companyGridObjectGridAttr.xml` eine Methode zum Konfigurieren der Attribute eines Element "objectGrid". Die Sicherheit wird aktiviert, der Berechtigungsmechanismus wird auf "JAAS" gesetzt, und

das Intervall für die Berechtigungsprüfung wird auf 45 Sekunden gesetzt. Außerdem registriert die Datei Entitäten, weil ein Attribut ″entityMetadataXMLFile″ angegeben wird.

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

 <objectGrids>
  <objectGrid name="CompanyGrid" securityEnabled="true"
   authorizationMechanism="AUTHORIZATION_MECHANISM_JASS"
   permissionCheckPeriod="45"
   entityMetadataXMLFile="companyGridEntities.xml">
   <backingMap name="Customer"/>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridObjectGridAttr.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);

companyGrid.setSecurityEnabled();
companyGrid.setAuthorizationMechanism(SecurityConstants.AUTHORIZATION_MECHANISM_JAAS);
companyGrid.setPermissionCheckPeriod(45);
companyGrid.registerEntities(new URL("file:companyGridEntities.xml"));
```

Element ″backingMap″

Das Element ″backingMap″ wird verwendet, um eine BackingMap-Instanz in einem ObjectGrid zu definieren. Jedes der Attribute im Element ″backingMap″ entspricht einer Methode in der Schnittstelle ″BackingMap″. Einzelheiten hierzu finden Sie im Abschnitt „Schnittstelle ″BackingMap″″ auf Seite 67.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: timeBasedDBUpdate

Attribute

name
Gibt den Namen an, der der backingMap-Instanz zugeordnet wird. Die XML-Validierung schlägt fehl, wenn dieses Attribut fehlt. (Erforderlich)

readOnly
Definiert eine BackingMap-Instanz mit Lese-/Schreibzugriff, wenn Sie den Wert `false` für das Attribut angeben. Wenn Sie den Wert `true` für das Attribut angeben, ist die BackingMap-Instanz schreibgeschützt. Aufrufe der Methode ″Session.getMap(String)″ führen zur Erstellung einer dynamischen Map, wenn der an die Methode übergebene Name dem regulären Ausdruck entspricht, der im Namensattribut dieser BackingMap angegeben ist. Der Standardwert ist `false`. (Optional)

template
Gibt an, ob dynamische Maps verwendet werden können. Setzen Sie dieses Attribut auf `true`, wenn die BackingMap-Map eine Schablonen-Map ist. Schablonen-Maps können verwendet werden, um Maps nach dem Start von ObjectGrid dynamisch zu starten. Aufrufe der Methode ″Session.getMap(String)″ führen zur Erstellung dynamischer Maps, wenn der an die Methode übergebene Name dem regulären Ausdruck entspricht, der im Namensattribut der BackingMap angegeben ist. Der Standardwert ist `false`. (Optional)

pluginCollectionRef
Gibt eine Referenz auf ein backingMapPluginCollection-Plug-in an. Der Wert dieses Attributs muss mit dem Attribut "ID" eines backingMapCollection-Plug-ins übereinstimmen. Die Validierung schlägt fehl, wenn keine übereinstimmende ID vorhanden ist. Setzen Sie das Attribut, um BackingMap-Plug-ins wiederzuverwenden. (Optional)

numberOfBuckets
Gibt die Anzahl der für die BackingMap-Instanz zu verwendenden Buckets an. Die BackingMap-Instanz verwendet eine Hash-Tabelle für ihre Implementierung. Wenn mehrere Einträge in der BackingMap vorhanden sind, kann durch mehr Buckets eine bessere Leistung erzielt werden, weil das Risiko von Kollisionen mit zunehmender Anzahl an Buckets sinkt. Werden zusätzliche Buckets verwendet, sind auch mehr gemeinsame Zugriffe möglich. Geben Sie den Wert `0` an, um den nahen Cache in einem Client zu inaktivieren, wenn die Kommunikation mit eXtreme Scale über Fernzugriff erfolgt. Wenn Sie das Attribut für einen Client auf `0` setzen, definieren Sie den Wert nur in der ObjectGrid-XML-Deskriptordatei für Clientkorrekturwerte. (Optional)

preloadMode
Legt den Modus für vorheriges Laden (Preload) fest, wenn ein Loader-Plug-in für diese BackingMap-Instanz definiert ist. Der Standardwert ist `false`. Wenn Sie das Attribut auf `true` setzen, wird die Methode "Loader.preloadMap(Session, BackingMap)" asynchron aufgerufen. Andernfalls wird die Methode beim Laden von Daten blockiert, so dass der Cache bis zum Abschluss des Preload-Prozesses nicht verfügbar ist. Der Preload-Prozess findet während der Initialisierung statt. (Optional)

lockStrategy
Gibt an, ob der interne Sperrenmanager verwendet wird, wenn eine Transaktion auf einen Map-Eintrag zugreift. Setzen Sie dieses Attribut auf einen der folgenden drei Werte: `OPTIMISTIC`, `PESSIMISTIC` oder `NONE`. Der Standardwert ist `OPTIMISTIC`. (Optional)

Die optimistische Sperrstrategie wird gewöhnlich verwendet, wenn eine Map kein Loader-Plug-in hat, wenn hauptsächlich Leseoperationen und weniger Schreib- und Aktualisierungsoperationen in der Map durchgeführt werden und wenn das Sperren nicht vom Persistenzmanager, der eXtreme Scale als Nebencache verwendet, oder von der Anwendung übernommen wird. Eine exklusive Sperre wird für einen Map-Eintrag angefordert, der bei der Festschreibung eingefügt, aktualisiert oder entfernt wird. Die Sperre stellt sicher, dass die Versionsinformationen von anderen Transaktionen nicht geändert werden können, während die Transaktion, die festgeschrieben wird, eine optimistische Versionsprüfung durchführt.

Die pessimistische Sperrstrategie wird gewöhnlich für eine Map verwendet, die kein Loader-Plug-in hat, und wenn das Sperren nicht von einem Persistenzmanager, der eXtreme Scale als Nebencache verwendet, von einem Loader-Plug-in oder von der Anwendung übernommen wird. Die pessimistische Sperrstrategie wird verwendet, wenn bei der optimistischen Sperrstrategie zu häufig Fehler auftreten, weil Aktualisierungstransaktionen für denselben Map-Eintrag zu häufig kollidieren.

Die Strategie ohne Sperren zeigt an, dass der interne Sperrenmanager nicht benötigt wird. Die Steuerung des gemeinsamen Zugriffs erfolgt außerhalb von eXtreme Scale durch den Persistenzmanager, der eXtreme Scale als Nebencache verwendet, durch die Anwendung oder durch das Loader-Plug-in, das Datenbanksperren für die Steuerung des gemeinsamen Zugriffs verwendet.

Weitere Informationen finden Sie im Abschnitt „Sperren von Map-Einträgen" auf Seite 71.

numberOfLockBuckets
Legt die Anzahl der Sperr-Buckets fest, die vom Sperrenmanager für die BackingMap-Instanz verwendet werden. Setzen Sie das Attribut "lockStrategy" auf `OPTIMISTIC` oder `PESSIMISTIC`, um einen Sperrenmanager für die BackingMap-Instanz zu erstellen. Der Sperrenmanager verwendet eine Hash-Tabelle, um die Einträge zu verfolgen, die von einer oder mehreren Transaktionen gesperrt werden. Wenn viele Einträge vorhanden sind, kann durch mehr Buckets eine bessere Leistung erzielt werden, weil das Risiko von Kollisionen mit zunehmender Anzahl an Buckets sinkt. Werden zusätzliche Buckets verwendet, sind auch mehr gemeinsame Zugriffe möglich. Setzen Sie das Attribut "lockStrategy" auf `NONE`, um anzugeben, dass die BackingMap-Instanz keinen Sperrenmanager verwendet. (Optional)

lockTimeout
Legt das Zeitlimit für Sperre fest, das vom Sperrenmanager für die BackingMap-Instanz verwendet wird. Setzen Sie das Attribut "lockStrategy" auf `OPTIMISTIC` oder `PESSIMISTIC`, um einen Sperrenmanager für die BackingMap-Instanz zu erstellen. Zur Vermeidung von Deadlocks hat der Sperrenmanager ein Standardzeitlimit von 15 Sekunden. Bei Überschreitung des Zeitlimits wird eine Ausnahme des Typs "LockTimeoutException" ausgelöst. Der Standardwert von 15 Sekunden ist für die meisten Anwendungen ausreichend, aber auf einem System mit hoher Belastung können Zeitlimitüberschreitungen auch dann auftreten, wenn kein Deadlock vorhanden ist. Verwenden Sie das Attribut "lockTimeout", um einen höheren als den Standardwert festzulegen, um falsche Zeitlimitüberschreitungsausnahmen zu verhindern. Setzen Sie das Attribut "lockStrategy" auf `NONE`, um anzugeben, dass die BackingMap-Instanz keinen Sperrenmanager verwendet. (Optional)

CopyMode
Gibt an, ob eine get-Operation eines Eintrags in der BackingMap-Instanz den tatsächlichen Wert, eine Kopie des Werts oder einen Proxy für den Wert zurückgibt. Setzen Sie das Attribut "CopyMode" auf einen der folgenden fünf Werte:

`COPY_ON_READ_AND_COMMIT`
Der Standardwert ist `COPY_ON_READ_AND_COMMIT`. Setzen Sie das Attribut auf `COPY_ON_READ_AND_COMMIT`, um sicherzustellen, dass eine Anwendung keine Referenz auf das Wertobjekt verwendet, das in der BackingMap-Instanz enthalten ist. Stattdessen arbeitet die Anwendung immer mit einer Kopie des Werts, der in der BackingMap-Instanz enthalten ist. (Optional)

`COPY_ON_READ`
Setzen Sie das Attribut auf `COPY_ON_READ`, um eine im Vergleich mit dem Wert `COPY_ON_READ_AND_COMMIT` höhere Leistung zu erzielen, indem der Kopiervorgang bei der Festschreibung einer Transaktion umgangen wird. Zur Gewährleistung der Integrität der BackingMap-Daten stellt die Anwendung sicher, dass jede Referenz auf einen Eintrag nach der Festschreibung der Transaktion gelöscht wird. Durch die Festlegung dieses Werts wird eine Methode "ObjectMap.get" aufgerufen, die eine Kopie des Werts an Stelle einer Referenz auf den Wert zurückgibt, was sicherstellt, dass Änderungen, die von der Anwendung am Wert vorgenommen werden, erst dann für das BackingMap-Element wirksam werden, wenn die Transaktion festgeschrieben wird.

COPY_ON_WRITE
Setzen Sie das Attribut auf COPY_ON_WRITE, um eine im Vergleich mit dem Wert COPY_ON_READ_AND_COMMIT bessere Leistung zu erzielen, in dem der Kopiervorgang beim ersten Aufruf der Methode "ObjectMap-.get" für einen bestimmten Schlüssel durch eine Transaktion umgangen wird. Stattdessen gibt die Methode "ObjectMap.get" einen Proxy für den Wert an Stelle einer Referenz auf das Wertobjekt zurück. Der Proxy stellt sicher, dass keine Kopie des Werts erstellt wird, sofern die Anwendung nicht eine Methode "set" für die Wertschnittstelle aufruft.

NO_COPY
Setzen Sie das Attribut auf den Wert NO_COPY, um einer Anwendung zu ermöglichen, ein Wertobjekt, das mit einer Methode "ObjectMap.get" abgerufen wird, nie zu ändern, um so Leistungsverbesserungen zu erzielen. Setzen Sie das Attribut für Maps, die Entitäten der EntityManager-API zugeordnet sind, auf NO_COPY.

COPY_TO_BYTES
Setzen Sie das Attribut auf COPY_TO_BYTES, um den Speicherbedarf für komplexe Objekttypen zu verringern und die Leistung zu verbessern, wenn das Kopieren des Objekts durch Serialisierung vorgenommen werden soll. Wenn ein Objekt nicht klonbar ist oder kein angepasstes ObjectTransformer-Plug-in mit einer effizienten Methode "copyValue" bereitgestellt wird, wird der Standardkopiermechanismus verwendet, um das Objekt zu serialisieren und zu dekomprimieren und die Kopie zu erstellen. Bei der Einstellung COPY_TO_BYTES wird die Dekomprimierung bei einer reinen Leseoperation und die Serialisierung bei einer reinen Festschreibungsoperation durchgeführt.

Weitere Informationen zu diesen Einstellungen finden Sie in den Informationen zu den bewährten Verfahren für CopyMode im *Programmierhandbuch*..

valueInterfaceClassName
Gibt eine Klasse an, die erforderlich ist, wenn Sie das Attribut "CopyMode" auf COPY_ON_WRITE setzen. Dieses Attribut wird für alle anderen Modi ignoriert. Beim Wert COPY_ON_WRITE wird ein Proxy verwendet, wenn die Methode "ObjectMap.get" aufgerufen wird. Der Proxy stellt sicher, dass keine Kopie des Werts erstellt wird, sofern die Anwendung keine set-Methode in der Klasse aufruft, die mit dem Attribut "valueInterfaceClassName" angegeben ist. (Optional)

copyKey
Gibt an, ob eine Kopie des Schlüssels erforderlich ist, wenn ein Map-Eintrag erstellt wird. Wenn das Schlüsselobjekt kopiert wird, kann die Anwendung dasselbe Schlüsselobjekt für jede ObjectMap-Operation verwenden. Setzen Sie das Attribut auf true, um das Schlüsselobjekt zu kopieren, wenn ein Map-Eintrag erstellt wird. Der Standardwert ist false. (Optional)

nullValuesSupported
Setzen Sie das Attribut auf true, um Nullwerte in der ObjectMap zu unterstützen. Wenn Nullwerte unterstützt werden, bedeutet eine get-Operation, die null zurückgibt, dass der Wert null ist oder dass die Map den an die Methode übergebenen Schlüssel nicht enthält. Der Standardwert ist true. (Optional)

ttlEvictorType
Gibt an, wie die Verfallszeit eines BackingMap-Eintrags berechnet wird. Setzen Sie dieses Attribut auf einen der folgenden drei Werte: CREATION_TIME, LAST_ACCESS_TIME oder NONE. Der Wert CREATION_TIME zeigt an, dass die Verfallszeit eines Eintrags die Summe aus Erstellungszeit des Eintrags und Wert

des Attributs ″timeToLive″ ist. Der Wert LAST_ACCESS_TIME zeigt an, dass die Verfallszeit eines Eintrags die Summe aus letzter Zugriffszeit des Eintrags und Wert des Attributs ″timeToLive″ ist. Der Wert NONE (der Standardwert) zeigt an, dass ein Eintrag keine Verfallszeit hat und so lange in der BackingMap-Instanz bleibt, bis die Anwendung den Eintrag explizit entfernt oder ungültig macht. (Optional)

timeToLive
Gibt an, wie lange (in Sekunden) jeder Map-Eintrag existiert. Der Standardwert 0 bedeutet, dass der Map-Eintrag unbegrenzt bzw. so lange existiert, bis die Anwendung den Eintrag explizit entfernt oder ungültig macht. Andernfalls entfernt das TTL-Bereinigungsprogramm den Map-Eintrag auf der Basis dieses Werts. (Optional)

streamRef
Gibt an, dass die BackingMap die Quellen-Map eines Datenstroms ist. Alle Einfüge- oder Aktualisierungsoperationen für die BackingMap werden in ein Streaming-Ereignis an die Abfragesteuerkomponente für Datenströme konvertiert. Dieses Attribut muss auf einen gültigen Datenstromnamen in einem streamQuerySet-Objekt verweisen. (Optional)

viewRef
Gibt an, dass die BackingMap eine Sicht-Map ist. Die Sichtausgabe der Abfragesteuerkomponente für Datenströme wird in das eXtreme-Scale-Tupelformat konvertiert und in der Map gespeichert. (Optional)

evictionTriggers
Legt die Typen zusätzlich zu verwendender Bereinigungs-Trigger fest. Alle Bereinigungsprogramme (Evictor) für die BackingMap verwenden diese liste zusätzlicher Trigger. Zur Vermeidung von Ausnahmen des Typs ″IllegalStateException″ muss dieses Attribut vor der Methode ″ObjectGrid.initialize()″ aufgerufen werden. Beachten Sie auch, dass die Methode ″ObjectGrid.getSession()″ die Methode ″ObjectGrid.initialize()″ implizit aufruft, falls die Methode noch nicht von der Anwendung aufgerufen wurde. Einträge in der Trigger-Liste werden durch Semikolons getrennt. Zu den aktuellen Bereinigungs-Triggern gehört MEMORY_USAGE_THRESHOLD. (Optional)

```
<backingMap
(1)   name="objectGridName"
(2)   readOnly="true" | "false"
(3)      template="true" | "false"
(4)   pluginCollectionRef="reference to backingMapPluginCollection"
(5)   numberOfBuckets="number of buckets"
(6)   preloadMode="true" | "false"
(7)   lockStrategy="OPTIMISTIC" | "PESSIMISTIC" | "NONE"
(8)   numberOfLockBuckets="number of lock buckets"
(9)   lockTimeout="lock timeout"
(10)   copyMode="COPY_ON_READ_AND_COMMIT" | "COPY_ON_READ" | "COPY_ON_WRITE"
       | "NO_COPY" | "COPY_TO_BYTES"
(11)   valueInterfaceClassName="value interface class name"
(12)   copyKey="true" | "false"
(13)   nullValuesSupported="true" | "false"
(14)   ttlEvictorType="CREATION_TIME" | "LAST_ACCESS_TIME|NONE"
(15)   timeToLive="time to live"
(16)   streamRef="reference to a stream"
(17)   viewRef="reference to a view"
(18)   writeBehind="write-behind parameters"
(19)   evictionTriggers="MEMORY_USAGE_THRESHOLD"
/>
```

Im folgenden Beispiel wird die Datei companyGridBackingMapAttr.xml verwendet, um eine BackingMap-Beispielkonfiguration zu veranschaulichen.

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

```
 <objectGrids>
<objectGrid name="CompanyGrid">
    <backingMap name="Customer" readOnly="true"
     numberOfBuckets="641" preloadMode="false"
     lockStrategy="OPTIMISTIC" numberOfLockBuckets="409"
     lockTimeout="30" copyMode="COPY_ON_WRITE"
     valueInterfaceClassName="com.ibm.websphere.samples.objectgrid.CounterValueInterface"
     copyKey="true" nullValuesSupported="false"
     ttlEvictorType="LAST_ACCESS_TIME" timeToLive="3000"/>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Der folgende Mustercode demonstriert den programmgesteuerten Ansatz, um dieselbe Konfiguration zu erhalten wie mit der Datei companyGridBackingMapAttr.xml aus dem vorherigen Beispiel:

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);

BackingMap customerMap = companyGrid.defineMap("Customer");
customerMap.setReadOnly(true);
customerMap.setNumberOfBuckets(641);
customerMap.setPreloadMode(false);
customerMap.setLockStrategy(LockStrategy.OPTIMISTIC);
customerMap.setNumberOfLockBuckets(409);
customerMap.setLockTimeout(30);

// Wenn der Kopiermodus auf COPY_ON_WRITE gesetzt wird, ist eine Klasse "valueInterface" erforderlich.
customerMap.setCopyMode(CopyMode.COPY_ON_WRITE,
 com.ibm.websphere.samples.objectgrid.CounterValueInterface.class);
customerMap.setCopyKey(true);
customerMap.setNullValuesSupported(false);
customerMap.setTtlEvictorType(TTLType.LAST_ACCESS_TIME);
customerMap.setTimeToLive(3000); // Lebensdauer (TTL) auf 50 Minuten setzen
```

Element ″bean″

Verwenden Sie das Element ″bean″, um Plug-ins zu definieren. Sie können Plug-ins Instanzen des Elements ″objectGrid″ und ″BackingMap″ zuordnen.

- Anzahl der Vorkommen im Element ″objectGrid″: 0 bis viele
- Anzahl der Vorkommen im Element ″backingMapPluginCollection″: 0 bis viele
- Untergeordnetes Element: property

Attribute

id Gibt den Typ des zu erstellenden Plug-ins an. (Erforderlich)

Die gültigen Plug-ins für eine Bean, die ein untergeordnetes Element des Elements ″objectGrid″ ist, sind in der folgenden Liste aufgeführt:

- TransactionCallback-Plug-in
- ObjectGridEventListener-Plug-in
- SubjectSource-Plug-in
- MapAuthorization-Plug-in
- SubjectValidation-Plug-in

Die gültigen Plug-ins für eine Bean, die ein untergeordnetes Element des Elements ″backingMapPluginCollection″ ist, sind in der folgenden Liste aufgeführt:

- Loader-Plug-in
- ObjectTransformer-Plug-in
- OptimisticCallback-Plug-in
- Evictor-Plug-in

- MapEventListener-Plug-in
- MapIndex-Plug-in

className
Gibt den Namen der Klasse bzw. SpringBean an, die zum Erstellen des Plug-ins instanziert werden soll. Die Klasse muss die Schnittstelle für den Plug-in-Typ implementieren. Wenn Sie beispielsweises `ObjectGridEventListener` als Wert für das Attribut ″id″ angeben, muss der Wert des Attributs ″className″ auf eine Klasse verweisen, die die Schnittstelle ″ObjectGridEventListener″ implementiert. (Erforderlich)

```
<bean
(1) id="TransactionCallback" | "ObjectGridEventListener" |"SubjectSource" |
    "MapAuthorization" | "SubjectValidation" | "Loader" | "ObjectTransformer" |
    "OptimisticCallback" | "Evictor" | "MapEventListener" | "MapIndexPlugin"
(2) className="class name" | "(spring)bean name"
/>
```

Im folgenden Beispiel wird die Datei `companyGridBean.xml` verwendet, um zu veranschaulichen, wie Plug-ins mit dem Element ″bean″ konfiguriert werden. Ein ObjectGridEventListener-Plug-in wird dem ObjectGrid ″CompanyGrid″ hinzugefügt. Das Attribut ″className″ für dieses Element ″bean″ ist ″com.ibm.websphere-.objectgrid.plugins.builtins.TranPropListener″. Diese Klasse implementiert die Schnittstelle ″com.ibm.websphere.objectgrid.plugins.ObjectGridEventListener″.

Es wird auch ein BackingMap-Plug-in in der Datei `companyGridBean.xml` definiert. Ein Evictor-Plug-in wird der BackingMap-Instanz ″Customer″ hinzugefügt. Das die Bean-ID ″Evictor″ lautet, muss das Attribut ″className″ eine Klasse angeben, die die Schnittstelle ″com.ibm.websphere.objectgrid.plugins.Evictor″ implementiert. Die Klasse ″com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor″ implementiert diese Schnittstelle. Die BackingMap referenziert ihre Plug-ins über das Attribut ″pluginCollectionRef″.

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

 <objectGrids>
<objectGrid name="CompanyGrid">
   bean id="ObjectGridEventListener"
    className="com.ibm.websphere.objectgrid.plugins.builtins.TranPropListener"/>
   <backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
<backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor/>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridBean.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
TranPropListener tranPropListener = new TranPropListener();
companyGrid.addEventListener(tranPropListener);

BackingMap customerMap = companyGrid.defineMap("Customer");
Evictor lruEvictor = new com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor();
customerMap.setEvictor(lruEvictor);
```

Element ″property″

Verwenden Sie das Element ″property″, um Plug-ins Eigenschaften hinzuzufügen. Der Name der Eigenschaft muss einer set-Methode in der von der übergeordneten Bean referenzierten Klasse entsprechen.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

name
Der Name der Eigenschaft. Der Wert, der diesem Attribut zugeordnet wird, muss einer set-Methode in der Klasse entsprechen, die mit dem Attribut ″className″ der übergeordneten Bean angegeben wird. Wenn Sie beispielsweise das Attribut ″className″ der Bean auf `com.ibm.MyPlugin` setzen und der angegebene Name der Eigenschaft `size` lautet, muss die Klasse `com.ibm.MyPlugin` eine Methode ″setSize″ enthalten. (Erforderlich)

type
Gibt den Typ der Eigenschaft an. Der Typ wird an die set-Methode übergeben, die mit dem Attribut ″name″ angegeben wurde. Die gültigen Werte sind primitive Java-Typen, ihre java.lang-Gegenstücke und java.lang.String. Die Attribute ″name″ und ″type″ müssen einer Methodensignatur im Attribut ″className″ der Bean entsprechen. Wenn Sie beispielsweise den Namen `size` und den Typ `int` festlegen, muss eine Methode ″setSize(int)″ in der Klasse vorhanden sein, die mit dem Attribut ″className″ für die Bean angegeben wurde. (Erforderlich)

value
Gibt den Wert der Eigenschaft an. Dieser Wert wird in den mit dem Attribut ″type″ angegebenen Typ konvertiert und anschließend als Parameter in dem Aufruf der set-Methode verwendet, die mit den Attributen ″name″ und ″type″ angegeben wurde. Der Wert dieses Attributs wird nicht validiert. (Erforderlich)

description
Beschreibt die Eigenschaft. (Optional)

```
<bean
(1)   name="name"
(2) type="java.lang.String" | "boolean" | "java.lang.Boolean" | "int" |
       "java.lang.Integer" | "double" | "java.lang.Double" | "byte" |
       "java.lang.Byte" | "short" | "java.lang.Short" | "long" |
       "java.lang.Long" | "float" | "java.lang.Float" | "char" |
       "java.lang.Character"
(3)   value="value"
(4) description="description"
/>
```

Im folgenden Beispiel wird die Datei `companyGridProperty.xml` verwendet, um zu veranschaulichen, wie einer Bean ein Element ″property″ hinzugefügt wird. In diesem Beispiel wird einem Bereinigungsprogramm (Evictor) eine Eigenschaft mit dem Namen ″maxSize″ und dem Typ ″int″ hinzugefügt. Das Bereinigungsprogramm ″com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor″ hat eine Methodensignatur, die der Methode ″setMaxSize(int)″ entspricht. Der ganzzahlige Wert 499 wird an die Methode ″setMaxSize(int)″ in der Klasse ″com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor″ übergeben.

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

 <objectGrids>
<objectGrid name="CompanyGrid">
```

```
   <backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
<backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor>
     <property name="maxSize" type="int" value="449"
       description="The maximum size of the LRU Evictor"/>
   </bean>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridProperty.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);

BackingMap customerMap = companyGrid.defineMap("Customer");

LRUEvictor lruEvictor = new com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor();
// Wenn Sie die XML-Datei verwenden würden, würde die hinzugefügte
// Eigenschaft den folgenden Aufruf bewirken
lruEvictor.setMaxSize(449);
customerMap.setEvictor(lruEvictor);
```

Element "backingMapPluginsCollections"

Das Element "backingMapPluginsCollections" ist ein Container für alle backingMapPluginCollection-Elemente. In der Datei `companyGridProperty.xml` aus dem vorherigen Beispiel enthält das Element "backingMapPluginCollections" ein einziges Element "backingMapPluginCollection" mit der ID `customerPlugins`.

- Anzahl der Vorkommen: 0 bis 1
- Untergeordnetes Element: backingMapPluginCollection

Element "backingMapPluginCollection"

Das Element "backingMapPluginCollection" definiert die BackingMap-Plug-ins und wird über das Attribut **id** identifiziert. Geben Sie das Attribut "pluginCollectionRef" an, um die Plug-ins zu referenzieren. Wenn Sie mehrere BackingMap-Plug-ins auf ähnliche Weise konfigurieren, kann jede BackingMap dasselbe Element "backingMapPluginCollection" referenzieren.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: bean

Attribute

id Identifiziert die backingMapPluginCollection und wird vom Attribut "pluginCollectionRef" des Elements "backingMap" referenziert. Jede ID muss eindeutig sein. Wenn der Wert des Attributs "pluginCollectionRef" nicht mit der ID eines der backingMapPluginCollection-Elemente übereinstimmt, schlägt die XML-Validierung fehl. Es können beliebig viele backingMap-Elemente ein einziges Element "backingMapPluginCollection" referenzieren. (Erforderlich)

```
<backingMapPluginCollection
(1)   id="id"
/>
```

Im folgenden Beispiel wird die Datei `companyGridCollection.xml` verwendet, um zu veranschaulichen, wie das Element "backingMapPluginCollection" verwendet wird. In dieser Datei verwendet die BackingMap "Customer" die backingMapPlug-

inCollection ″customerPlugins″, um die BackingMap ″Customer″ mit einem LRUEvictor-Plug-in zu konfigurieren. Die BackingMaps ″Item″ und ″OrderLine″ referenzieren die backingMapPluginCollection ″collection2″. Für diese BackingMaps ist jeweils ein LFUEvictor-Plug-in definiert.

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

 <objectGrids>
<objectGrid name="CompanyGrid">
   <backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
   <backingMap name="Item" pluginCollectionRef="collection2"/>
   <backingMap name="OrderLine"
    pluginCollectionRef="collection2"/>
<backingMap name="Order"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
<backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor/>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="collection2">
   <bean id="Evictor"
     className="com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor/>
   <bean id="OptimisticCallback"
     className="com.ibm.websphere.samples.objectgrid.EmployeeOptimisticCallBackImpl"/>
  </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridCollection.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();

ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
BackingMap customerMap = companyGrid .defineMap("Customer");
LRUEvictor customerEvictor = new LRUEvictor();
customerMap.setEvictor(customerEvictor);

BackingMap itemMap = companyGrid.defineMap("Item");
LFUEvictor itemEvictor = new LFUEvictor();
itemMap.setEvictor(itemEvictor);

BackingMap orderLineMap = companyGrid.defineMap("OrderLine");
LFUEvictor orderLineEvictor = new LFUEvictor();
orderLineMap.setEvictor(orderLineEvictor);

BackingMap orderMap = companyGrid.defineMap("Order");
```

Element ″querySchema″

Das Element ″querySchema″ definiert Beziehungen zwischen BackingMaps und identifiziert den Objekttyp jeder Map. Diese Informationen werden von ObjectQuery verwendet, um Zeichenfolgen in der Abfragesprache in Map-Zugriffsaufrufe zu übersetzen.

- Anzahl der Vorkommen: 0 bis 1
- Untergeordnete Elemente: mapSchemas, relationships

Element ″mapSchemas″

Jedes Element ″querySchema″ hat ein einziges Element ″mapSchemas″, das ein oder mehrere mapSchema-Elemente enthält.

- Anzahl der Vorkommen: 1

- Untergeordnetes Element: mapSchema

Element "mapSchema"

Ein Element "mapSchema" definiert den Typ der Objekte, die in einer BackingMap gespeichert werden, und enthält Anweisungen zum Zugriff auf die Daten.

- Anzahl der Vorkommen: 1 oder mehr
- Untergeordnetes Element: Ohne

Attribute

mapName
Gibt den Namen der BackingMap an, die dem Schema hinzugefügt werden soll. (Erforderlich)

valueClass
Gibt den Typ des Objekts an, der im Wertabschnitt der BackingMap gespeichert wird. (Erforderlich)

primaryKeyField
Gibt den Namen des Primärschlüsselattributs im Attribut "valueClass" an. Der Primärschlüssel muss auch im Schlüsselabschnitt der BackingMap gespeichert werden. (Optional)

accessType
Gibt an, wie die Abfragesteuerkomponente die persistenten Daten in den valueClass-Objektinstanzen überwacht und auf diese zugreift. Wenn Sie das Attribut auf den Wert `FIELD` setzen, werden die Klassenfelder überwacht und dem Schema hinzugefügt. Wenn Sie das Attribut auf den Wert `PROPERTY` setzen, werden die Attribute, die den get- und is-Methoden zugeordnet sind, verwendet. Der Standardwert ist `PROPERTY`. (Optional)

```
<mapSchema
(1)   mapName="backingMapName"
(2)   valueClass="com.mycompany.OrderBean"
(3)   primaryKeyField="orderId"
(4)   accessType="PROPERTY" | "FIELD"
/>
```

Im folgenden Beispiel wird die Datei `companyGridQuerySchemaAttr.xml` verwendet, um eine Beispielkonfiguration für mapSchema zu veranschaulichen:

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

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

   <querySchema>
    <mapSchemas>
     <mapSchema mapName="Order"
      valueClass="com.mycompany.OrderBean"
      primaryKeyField="orderNumber"
      accessType="FIELD"/>
     <mapSchema mapName="Customer"
      valueClass="com.mycompany.CustomerBean"
      primaryKeyField="id"
      accessType="FIELD"/>
    </mapSchemas>
   </querySchema>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei companyGridQuerySchemaAttr.xml aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Schema definieren
QueryConfig queryCfg = new QueryConfig();
queryCfg.addQueryMapping(new QueryMapping(
    "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryMapping(new QueryMapping(
    "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
companyGrid.setQueryConfig(queryCfg);
```

Element ″relationships″

Jedes Element ″querySchema″ hat kein oder ein Element ″relationships″, das eine oder mehrere Elemente ″relationship″ enthält.

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: relationship

Element ″relationship″

Ein Element ″relationship″ definiert die Beziehung zwischen zwei BackingMaps und die Attribute im Attribut ″valueClass″, die für die Beziehungsbindung verwendet werden.

- Anzahl der Vorkommen: 1 oder mehr
- Untergeordnetes Element: Ohne

Attribute

source
Gibt den Namen der valueClass der Quellenseite einer Beziehung an. (Erforderlich)

target
Gibt den Namen der valueClass der Zielseite einer Beziehung an. (Erforderlich)

relationField
Gibt den Namen des Attributs in der Quellen-valueClass an, die auf das Ziel verweist. (Erforderlich)

invRelationField
Gibt den Namen des Attributs in der Ziel-valueClass an, die auf die Quelle verweist. Wenn Sie dieses Attribut nicht angeben, ist die Beziehung unidirektional. (Optional)

```
<mapSchema
(1)   source="com.mycompany.OrderBean"
(2)   target="com.mycompany.CustomerBean"
(3)   relationField="customer"
(4)   invRelationField="orders"
/>
```

Im folgenden Beispiel wird die Datei companyGridQuerySchemaWithRelationshipAttr.xml verwendet, um eine mapSchema-Musterkonfiguration zu veranschaulichen, die eine bidirektionale Beziehung enthält.

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

```
 xmlns="http://ibm.com/ws/objectgrid/config">

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

   <querySchema>
    <mapSchemas>
     <mapSchema mapName="Order"
      valueClass="com.mycompany.OrderBean"
      primaryKeyField="orderNumber"
      accessType="FIELD"/>
     <mapSchema mapName="Customer"
      valueClass="com.mycompany.CustomerBean"
      primaryKeyField="id"
      accessType="FIELD"/>
    </mapSchemas>
    <relationships>
     <relationship
      source="com.mycompany.OrderBean"
      target="com.mycompany.CustomerBean"
      relationField="customer"/>
      invRelationField="orders"/>
    </relationships>
   </querySchema>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridQuerySchemaWithRelationshipAttr.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Schema definieren
QueryConfig queryCfg = new QueryConfig();
queryCfg.addQueryMapping(new QueryMapping(
    "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryMapping(new QueryMapping(
    "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryRelationship(new QueryRelationship(
     OrderBean.class.getName(), CustomerBean.class.getName(), "customer", "orders"));
companyGrid.setQueryConfig(queryCfg);
```

Element ″streamQuerySet″

Das Element ″streamQuerySet″ ist das Ausgangselement für die Definition eines Abfragesatzes für Datenströme.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnete Elemente: stream, view

Element ″stream″

Das Element ″stream″ stellt einen Datenstrom für die Abfragesteuerkomponente für Datenströme dar. Jedes Attribut des Elements ″stream″ entspricht einer Methode in der Schnittstelle ″StreamMetadata″.

- Anzahl der Vorkommen: 1 bis viele
- Untergeordnetes Element: basic

Attribute

name
Gibt den Namen des Datenstroms an. Die Validierung schlägt fehl, wenn dieses Attribut nicht angegeben wird. (Erforderlich)

valueClass
Gibt den Klassentyp des Werts an, der in der ObjectMap des Datenstroms gespeichert wird. Der Klassentyp wird verwendet, um das Objekt in Datenstromereignisse zu konvertieren und um eine SQL-Anweisung zu generieren, wenn die Anweisung nicht angegeben ist. (Erforderlich)

sql
Gibt die SQL-Anweisung des Datenstroms an. Wenn Sie diese Eigenschaft nicht angeben, wird eine Datenstrom-SQL durch Reflexion der Attribute bzw. Zugriffsmethoden im Attribut ″valueClass″ bzw. durch Verwendung der Tupelattribute der Entitätsmetadaten generiert. (Optional)

access
Gibt den Typ für den Zugriff auf die Attribute der Wertklasse an. Wenn Sie das Attribut auf den Wert `FIELD` setzen, werden die Attribute durch Java-Reflexion direkt aus den Feldern abgerufen. Andernfalls werden Zugriffsmethoden für das Lesen der Attribute verwendet. Der Standardwert ist `PROPERTY`. (Optional)

```
<stream
(1)   name="streamName"
(2)   valueClass="streamMapClassType"
(3)   sql="streamSQL create stream stockQuote
     keyed by t ( transactionvolume INTEGER, price DECIMAL (9,2), issue VARCHAR(100) );"
(4)   access="PROPERTY" | "FIELD"
/>
```

Element ″view″

Das Element ″view″ stellt eine Datenstromabfragesicht dar. Jedes Element ″stream″ entspricht einer Methode in der Schnittstelle ″ViewMetadata″.

- Anzahl der Vorkommen: 1 bis viele
- Untergeordnete Elemente: basic, id

Attribute

name
Gibt den Namen der Sicht an. Die Validierung schlägt fehl, wenn dieses Attribut nicht angegeben wird. (Erforderlich)

sql
Gibt die SQL des Datenstroms an, die die Sichtkonvertierung definiert. Die Validierung schlägt fehl, wenn dieses Attribut nicht angegeben wird. (Erforderlich)

valueClass
Gibt den Klassentyp des Werts an, der in dieser Sicht der ObjectMap gespeichert wird. Der Klassentyp wird verwendet, um die Sichtereignisse in das richtige Tupelformat zu konvertieren, das mit diesem Klassentyp kompatibel ist. Wenn Sie den Klassentyp nicht angeben, wird ein Standardformat gemäß den Spaltendefinition in Stream Processing Technology Structured Query Language (SPTSQL) verwendet. Wenn Entitätsmetadaten für diese Sicht-Map definiert sind, darf dieses Attribut nicht verwendet werden. Stattdessen werden die Entitätsmetadaten verwendet. (Optional)

access
Gibt den Typ für den Zugriff auf die Attribute der Wertklasse an. Wenn Sie den Zugriffstyp auf `FIELD` setzen, werden die Spaltenwerte durch Java-Refle-

xion direkt aus den Feldern abgerufen. Andernfalls werden Zugriffsmethoden für das Definieren der Attribute verwendet. Der Standardwert ist PROPERTY. (Optional)

```
<view
(1)   name="viewName"
(2)   valueClass="viewMapValueClass"
(3)   sql="viewSQL CREATE VIEW last5MinuteAvgPrice AS
      SELECT issue, avg(price) as totalVolume
        FROM (SELECT * FROM stockQuote FETCH LATEST 5 MINUTES) group by issue;"/>
(4)   access="PROPERTY" | "FIELD"
/>
```

Element "basic"

Das Element "basic" wird verwendet, um eine Zuordnung des Attributnamens in der Wertklasse bzw. den Entitätsmetadaten zu der in SPTSQL definierten Spalte zu definieren.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

```
<basic
(1)   name="attributeName"
(2)   column="columnName"
/>
```

Element "id"

Das Element "id" wird für die Zuordnung eines Schlüsselattributs verwendet.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

```
<id
(1)   name="idName"
(2)   column="columnName"
/>
```

Im folgenden Beispiel wird die Datei StreamQueryApp2.xml verwendet, um zu veranschaulichen, wie die Attribut eines Datenstromabfragesatzes konfiguriert werden. Der Datenstromabfragesatz "_stockQuoteSQS_" hat einen Datenstrom und eine Sicht. Datenstrom und Sicht definieren einen Namen, die Wertklasse, die SQL und den Zugriffstyp. Der Datenstrom definiert auch ein Element "basic", das angibt, dass das Attribut "volume" in der Klasse "StockQuote" der in der SQL-Anweisung definierten SQL-Spalte "transactionvolume" zugeordnet ist.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
 <objectGrids>
  <objectGrid name="og1">
   <backingMap name="stockQuote" readOnly="false" copyKey="true" streamRef="stockQuote"/>
   <backingMap name="last5MinuteAvgPrice" readOnly="false" copyKey="false" viewRef="last5MinuteAvgPrice"/>

   <streamQuerySet name="stockQuoteSQS">
    <stream
     name="stockQuote"
     valueClass="com.ibm.ws.objectgrid.streamquery.sample.guide.StockQuote"
     sql="create stream stockQuote
         keyed by t ( transactionvolume INTEGER, price DECIMAL (9,2), issue VARCHAR(100) );"
     access="FIELD">
     <basic name="volume" column="transactionvolume"/>
    </stream>

    <view
     name="last5MinuteAvgPrice"
     valueClass="com.ibm.ws.objectgrid.streamquery.sample.guide.AveragePrice"
     sql="CREATE VIEW last5MinuteAvgPrice AS SELECT issue, avg(price) as avgPrice
```

```
      FROM (SELECT * FROM stockQuote FETCH LATEST 5 MINUTES) group by issue;"
     access="FIELD"
    </view>
   </streamQuerySet>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Datei objectGrid.xsd

Verwenden Sie das ObjectGrid-XML-Deskriptorschema, um WebSphere eXtreme Scale zu konfigurieren.

Im Abschnitt „ObjectGrid-XML-Deskriptordatei" auf Seite 157 finden Sie Beschreibungen der Elemente und Attribute, die in der Datei `objectGrid.xsd` definiert sind. Informationen zur Spring-Datei `objectgrid.xsd` finden Sie im Abschnitt „Spring-XML-Deskriptordatei" auf Seite 216.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:cc="http://ibm.com/ws/objectgrid/config"
xmlns:dgc="http://ibm.com/ws/objectgrid/config"
elementFormDefault="qualified"
targetNamespace="http://ibm.com/ws/objectgrid/config">

 <xsd:element name="objectGridConfig">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element maxOccurs="1" minOccurs="1" name="objectGrids"
       type="dgc:objectGrids">
     <xsd:unique name="objectGridNameUnique">
      <xsd:selector xpath="dgc:objectGrid"/>
      <xsd:field xpath="@name"/>
     </xsd:unique>
    </xsd:element>
    <xsd:element maxOccurs="1" minOccurs="0" name="backingMapPluginCollections"
       type="dgc:backingMapPluginCollections"/>
   </xsd:sequence>
  </xsd:complexType>

  <xsd:key name="backingMapPluginCollectionId">
   <xsd:selector xpath="dgc:backingMapPluginCollections/dgc:
     backingMapPluginCollection"/>
   <xsd:field xpath="@id"/>
  </xsd:key>

  <xsd:keyref name="pluginCollectionRef" refer="dgc:backingMapPluginCollectionId">
   <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:backingMap"/>
   <xsd:field xpath="@pluginCollectionRef"/>
  </xsd:keyref>

  <xsd:key name="streamName">
   <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:
     streamQuerySet/dgc:stream"/>
   <xsd:field xpath="@name"/>
  </xsd:key>

  <xsd:keyref name="streamRef" refer="dgc:streamName">
   <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:backingMap"/>
   <xsd:field xpath="@streamRef"/>
  </xsd:keyref>

  <xsd:key name="viewName">
   <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:streamQuerySet/dgc:view"/>
   <xsd:field xpath="@name"/>
  </xsd:key>

  <xsd:keyref name="viewRef" refer="dgc:viewName">
   <xsd:selector xpath="dgc:objectGrids/dgc:objectGrid/dgc:backingMap"/>
   <xsd:field xpath="@viewRef"/>
  </xsd:keyref>
 </xsd:element>

 <xsd:complexType name="objectGrids">
  <xsd:sequence>
   <xsd:element maxOccurs="unbounded" minOccurs="1" name="objectGrid"
      type="dgc:objectGrid">
```

```
   <xsd:unique name="backingMapNameUnique">
    <xsd:selector xpath="dgc:backingMap"/>
    <xsd:field xpath="@name"/>
   </xsd:unique>
   <xsd:unique name="streamQuerySetNameUnique">
    <xsd:selector xpath="dgc:streamQuerySet"/>
    <xsd:field xpath="@name"/>
   </xsd:unique>
  </xsd:element>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="backingMapPluginCollections">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="backingMapPluginCollection"
    type="dgc:backingMapPluginCollection"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="objectGrid">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="bean" type="dgc:bean"/>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="backingMap"
     type="dgc:backingMap"/>
  <xsd:element maxOccurs="1" minOccurs="0" name="querySchema" type="dgc:querySchema"/>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="streamQuerySet"
     type="dgc:streamQuerySet">
   <xsd:unique name="stream">
    <xsd:selector xpath="dgc:stream"/>
    <xsd:field xpath="@name"/>
   </xsd:unique>
   <xsd:unique name="view">
    <xsd:selector xpath="dgc:view"/>
    <xsd:field xpath="@name"/>
   </xsd:unique>
  </xsd:element>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="authorizationMechanism" type="dgc:authorizationMechanism"
   use="optional"/>
 <xsd:attribute name="accessByCreatorOnlyMode" type="dgc:accessByCreatorOnlyMode"
   use="optional"/>
 <xsd:attribute name="securityEnabled" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="txTimeout" type="xsd:int" use="optional"/>
 <xsd:attribute name="permissionCheckPeriod" type="xsd:int" use="optional"/>
 <xsd:attribute name="entityMetadataXMLFile" type="xsd:string" use="optional"/>
 <xsd:attribute name="initialState" type="dgc:initialState" use="optional"/>
</xsd:complexType>


<xsd:complexType name="backingMap">
 <xsd:sequence>
  <xsd:element maxOccurs="1" minOccurs="0" name="timeBasedDBUpdate" type="dgc:
    timeBasedDBUpdate"/>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="readOnly" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="pluginCollectionRef" type="xsd:string" use="optional"/>
 <xsd:attribute name="preloadMode" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="lockStrategy" type="dgc:lockStrategy" use="optional"/>
 <xsd:attribute name="copyMode" type="dgc:copyMode" use="optional"/>
 <xsd:attribute name="valueInterfaceClassName" type="xsd:string" use="optional"/>
 <xsd:attribute name="numberOfBuckets" type="xsd:int" use="optional"/>
 <xsd:attribute name="nullValuesSupported" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="lockTimeout" type="xsd:int" use="optional"/>
 <xsd:attribute name="numberOfLockBuckets" type="xsd:int" use="optional"/>
 <xsd:attribute name="copyKey" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="timeToLive" type="xsd:int" use="optional"/>
 <xsd:attribute name="ttlEvictoryType" type="dgc:ttlEvictorType" use="optional"/>
 <xsd:attribute name="streamRef" type="xsd:string" use="optional"/>
 <xsd:attribute name="viewRef" type="xsd:string" use="optional"/>
 <xsd:attribute name="writeBehind" type="xsd:string" use="optional"/>
 <xsd:attribute name="evictionTriggers" type="xsd:string" use="optional"/>
 <xsd:attribute name="template" type="xsd:boolean" use="optional"/>
</xsd:complexType>

<xsd:complexType name="bean">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="property" type="dgc:property"/>
```

```
 </xsd:sequence>
 <xsd:attribute name="className" type="xsd:string" use="required"/>
 <xsd:attribute name="id" type="dgc:beanId" use="required"/>
</xsd:complexType>

<xsd:complexType name="backingMapPluginCollection">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="bean" type="dgc:bean"/>
 </xsd:sequence>
 <xsd:attribute name="id" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="property">
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="value" type="xsd:string" use="required"/>
 <xsd:attribute name="type" type="dgc:propertyType" use="required"/>
 <xsd:attribute name="description" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:simpleType name="propertyType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="java.lang.Boolean"/>
  <xsd:enumeration value="boolean"/>
  <xsd:enumeration value="java.lang.String"/>
  <xsd:enumeration value="java.lang.Integer"/>
  <xsd:enumeration value="int"/>
  <xsd:enumeration value="java.lang.Double"/>
  <xsd:enumeration value="double"/>
  <xsd:enumeration value="java.lang.Byte"/>
  <xsd:enumeration value="byte"/>
  <xsd:enumeration value="java.lang.Short"/>
  <xsd:enumeration value="short"/>
  <xsd:enumeration value="java.lang.Long"/>
  <xsd:enumeration value="long"/>
  <xsd:enumeration value="java.lang.Float"/>
  <xsd:enumeration value="float"/>
  <xsd:enumeration value="java.lang.Character"/>
  <xsd:enumeration value="char"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="beanId">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="TransactionCallBack"/>
  <xsd:enumeration value="ObjectGridEventListener"/>
  <xsd:enumeration value="SubjectSource"/>
  <xsd:enumeration value="MapAuthorization"/>
  <xsd:enumeration value="SubjectValidation"/>
  <xsd:enumeration value="ObjectGridAuthorization"/>

  <xsd:enumeration value="Loader"/>
  <xsd:enumeration value="ObjectTransformer"/>
  <xsd:enumeration value="OptimisticCallback"/>
  <xsd:enumeration value="Evictor"/>
  <xsd:enumeration value="MapEventListener"/>
  <xsd:enumeration value="MapIndexPlugin"/>

 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="copyMode">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="COPY_ON_READ_AND_COMMIT"/>
  <xsd:enumeration value="COPY_ON_READ"/>
  <xsd:enumeration value="COPY_ON_WRITE"/>
  <xsd:enumeration value="NO_COPY"/>
  <xsd:enumeration value="COPY_TO_BYTES"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="lockStrategy">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="OPTIMISTIC"/>
  <xsd:enumeration value="PESSIMISTIC"/>
  <xsd:enumeration value="NONE"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="ttlEvictorType">
```

```
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="CREATION_TIME"/>
  <xsd:enumeration value="LAST_ACCESS_TIME"/>
  <xsd:enumeration value="NONE"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="authorizationMechanism">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="AUTHORIZATION_MECHANISM_JAAS"/>
  <xsd:enumeration value="AUTHORIZATION_MECHANISM_CUSTOM"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="accessByCreatorOnlyMode">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="disabled"/>
  <xsd:enumeration value="complement"/>
  <xsd:enumeration value="supersede"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="streamQuerySet">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="stream" type="dgc:stream">
   <xsd:unique name="streamBasicColumnUnique">
    <xsd:selector xpath="dgc:basic"/>
    <xsd:field xpath="@column"/>
   </xsd:unique>
  </xsd:element>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="view" type="dgc:view">
   <xsd:unique name="viewBasicColumnUnique">
    <xsd:selector xpath="dgc:basic"/>
    <xsd:field xpath="@column"/>
   </xsd:unique>
  </xsd:element>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="viewResultsToListenersOnly" type="xsd:boolean"
    default="false" use="optional"/>
 <xsd:attribute name="deployInPrimaryOnly" type="xsd:boolean" default="true"
    use="optional"/>
</xsd:complexType>

<xsd:complexType name="stream">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="basic" type="dgc:basic"/>
 </xsd:sequence>
 <xsd:attribute name="valueClass" type="xsd:string" use="required"/>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="sql" type="xsd:string" use="optional"/>
 <xsd:attribute name="access" type="cc:accessType" use="optional"/>
</xsd:complexType>

<xsd:complexType name="view">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="0" name="id" type="dgc:basic"/>
  <xsd:element element maxOccurs="unbounded" minOccurs="0" name="basic"
    type="dgc:basic"/>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="sql" type="xsd:string" use="optional"/>
 <xsd:attribute name="valueClass" type="xsd:string" use="optional"/>
 <xsd:attribute name="access" type="cc:accessType" use="optional"/>
</xsd:complexType>

<xsd:complexType name="basic">
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="column" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="id">
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="column" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="timeBasedDBUpdate">
 <xsd:attribute name="persistenceUnitName" type="xsd:string" use="optional"/>
 <xsd:attribute name="mode" type="cc:dbUpdateMode" use="optional"/>
```

```
 <xsd:attribute name="timestampField" type="xsd:string" use="optional"/>
 <xsd:attribute name="entityClass" type="xsd:string" use="required"/>
 <xsd:attribute name="jpaPropertyFactory" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:simpleType name="dbUpdateMode">
 <xsd:restriction base="xsd:string"/>
  <xsd:enumeration value="INVALIDATE_ONLY"/>
  <xsd:enumeration value="UPDATE_ONLY"/>
  <xsd:enumeration value="INSERT_UPDATE"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="querySchema">
 <xsd:sequence>
  <xsd:element maxOccurs="1" minOccurs="1" name="mapSchemas" type="dgc:mapSchemas">
   <xsd:unique name="mapNameUnique">
    <xsd:selector xpath="dgc:mapSchema"/>
    <xsd:field xpath="@mapName"/>
   </xsd:unique>
  </xsd:element>
  <xsd:element maxOccurs="1" minOccurs="0" name="relationships"
    type="dgc:relationships"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="mapSchemas">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="1" name="mapSchema"
    type="dgc:mapSchema"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="relationships">
 <xsd:sequence>
  <xsd:element maxOccurs="unbounded" minOccurs="1" name="relationship"
    type="dgc:relationship"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="mapSchema">
 <xsd:attribute name="mapName" type="xsd:string" use="required"/>
 <xsd:attribute name="valueClass" type="xsd:string" use="required"/>
 <xsd:attribute name="primaryKeyField" type="xsd:string" use="optional"/>
 <xsd:attribute name="accessType" type="cc:accessType" use="optional"/>
</xsd:complexType>

<xsd:complexType name="relationship">
 <xsd:attribute name="source" type="xsd:string" use="required"/>
 <xsd:attribute name="target" type="xsd:string" use="required"/>
 <xsd:attribute name="relationField" type="xsd:string" use="required"/>
 <xsd:attribute name="invRelationField" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:simpleType name="accessType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="PROPERTY"/>
  <xsd:enumeration value="FIELD"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="initialState">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="OFFLINE"/>
  <xsd:enumeration value="PRELOAD"/>
  <xsd:enumeration value="ONLINE"/>
 </xsd:restriction>
</xsd:simpleType>

</xsd:schema>
```

XML-Deskriptordatei für Entitätsmetadaten

Die Deskriptordatei für Entitätsmetadaten ist eine XML-Datei, die verwendet wird, um ein Entitätsschema für WebSphere eXtreme Scale zu definieren. Definieren Sie alle Entitätsmetadaten in der XML-Datei, oder definieren Sie die Entitätsmetadaten

als Annotationen in der Java-Klassendatei der Entität. Vorrangig wird die XML-Deskriptordatei für Entitäten verwendet, die keine Java-Annotationen verwenden können.

Verwenden Sie XML-Konfiguration, um Entitätsmetadaten zu erstellen, die auf der XML-Datei basieren. Wenn die XML-Konfiguration in Kombination mit Annotationen verwendet wird, überschreiben einige der Attribute, die in der XML-Konfiguration definiert werden, die entsprechenden Annotationen. Wenn Sie ein Element überschreiben, wird dies explizit in den folgenden Abschnitten hervorgehoben. Ein Beispiel für die XML-Deskriptordatei für Entitätsmetadaten finden Sie im Abschnitt „Datei `emd.xsd`" auf Seite 189.

Element "id"

Das Element "id" impliziert, dass das Attribut ein Schlüssel ist. Es muss mindestens ein Element "id" angegeben werden. Sie können mehrere id-Schlüssel als Verbundschlüssel angeben.

Attribute

name
Gibt den Namen des Attributs an. Das Attribut muss in der Java-Datei enthalten sein.

alias
Gibt den Elementalias an. Der Aliaswert wird überschrieben, wenn er zusammen mit einer annotierten Entität verwendet wird.

Element "basic"

Das Element "basic" impliziert, dass das Attribut ein primitiver Typ oder ein Wrapper für primitive Typen ist.

- java.lang.String
- java.math.BigInteger
- java.math.BigDecimal
- java.util.Date
- java.util.Calendar
- java.sql.Date
- java.sql.Time
- java.sql.Timestamp
- byte[]
- Byte[]
- char[]
- Character[]
- Java Platform, Standard Edition Version 5 enum

Es ist nicht erforderlich, Attribute für "basic" anzugeben. Die Attribute des Elements "basic" werden automatisch über Reflexion konfiguriert.

Element "id-class"

Das Element "id_class" gibt eine Verbundschlüsselklasse an, mit der Entitäten über Verbundschlüssel gesucht werden können.

Attribute

class-name
Gibt den Klassennamen (eine id-Klasse) an, der für das Element ″id-class″ verwendet werden soll.

transient

Das Element ″transient″ impliziert, dass es ignoriert und nicht verarbeitet wird. Es kann auch überschrieben werden, wenn es zusammen mit annotierten Entitäten verwendet wird.

Attribute

name
Gibt den Namen des Attributs an, das ignoriert wird.

version

Das Element ″transient″ impliziert, dass es ignoriert und nicht verarbeitet wird. Es kann auch überschrieben werden, wenn es zusammen mit annotierten Entitäten verwendet wird.

Attribute

name
Gibt den Namen des Attributs an, das ignoriert wird.

Element ″property″

Verwenden Sie das Element ″property″, um Plug-ins Eigenschaften hinzuzufügen. Der Name der Eigenschaft muss einer set-Methode in der von der übergeordneten Bean referenzierten Klasse entsprechen.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

name
Der Name der Eigenschaft. Der Wert, der diesem Attribut zugeordnet wird, muss einer set-Methode in der Klasse entsprechen, die mit dem Attribut ″class-Name″ der übergeordneten Bean angegeben wird. Wenn Sie beispielsweise das Attribut ″className″ der Bean auf `com.ibm.MyPlugin` setzen und der angegebene Name der Eigenschaft `size` lautet, muss die Klasse `com.ibm.MyPlugin` eine Methode ″setSize″ enthalten. (Erforderlich)

type
Gibt den Typ der Eigenschaft an. Der Typ wird an die set-Methode übergeben, die mit dem Attribut ″name″ angegeben wurde. Die gültigen Werte sind primitive Java-Typen, ihre java.lang-Gegenstücke und java.lang.String. Die Attribute ″name″ und ″type″ müssen einer Methodensignatur im Attribut ″className″ der Bean entsprechen. Wenn Sie beispielsweise den Namen `size` und den Typ `int` festlegen, muss eine Methode ″setSize(int)″ in der Klasse vorhanden sein, die mit dem Attribut ″className″ für die Bean angegeben wurde. (Erforderlich)

value
Gibt den Wert der Eigenschaft an. Dieser Wert wird in den mit dem Attribut

″type″ angegebenen Typ konvertiert und anschließend als Parameter in dem Aufruf der set-Methode verwendet, die mit den Attributen ″name″ und ″type″ angegeben wurde. Der Wert dieses Attributs wird nicht validiert. (Erforderlich)

description
Beschreibt die Eigenschaft. (Optional)

```
<bean
(1)   name="name"
(2)   type="java.lang.String" | "boolean" | "java.lang.Boolean" | "int" |
       "java.lang.Integer" | "double" | "java.lang.Double" | "byte" |
       "java.lang.Byte" | "short" | "java.lang.Short" | "long" |
       "java.lang.Long" | "float" | "java.lang.Float" | "char" |
       "java.lang.Character"
(3)   value="value"
(4)   description="description"
/>
```

Im folgenden Beispiel wird die Datei `companyGridProperty.xml` verwendet, um zu veranschaulichen, wie einer Bean ein Element ″property″ hinzugefügt wird. In diesem Beispiel wird einem Bereinigungsprogramm (Evictor) eine Eigenschaft mit dem Namen ″maxSize″ und dem Typ ″int″ hinzugefügt. Das Bereinigungsprogramm ″com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor″ hat eine Methodensignatur, die der Methode ″setMaxSize(int)″ entspricht. Der ganzzahlige Wert 499 wird an die Methode ″setMaxSize(int)″ in der Klasse ″com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor″ übergeben.

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

 <objectGrids>
<objectGrid name="CompanyGrid">
<backingMap name="Customer"
pluginCollectionRef="customerPlugins" />
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
<backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor>
     <property name="maxSize" type="int" value="449"
       description="The maximum size of the LRU Evictor"/>
   </bean>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridProperty.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);

BackingMap customerMap = companyGrid.defineMap("Customer");

LRUEvictor lruEvictor = new com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor();
// Wenn Sie die XML-Datei verwenden würden, würde die hinzugefügte
// Eigenschaft den folgenden Aufruf bewirken
lruEvictor.setMaxSize(449);
customerMap.setEvictor(lruEvictor);
```

Element ″backingMapPluginsCollections″

Das Element ″backingMapPluginsCollections″ ist ein Container für alle backingMapPluginCollection-Elemente. In der Datei `companyGridProperty.xml` aus

dem vorherigen Beispiel enthält das Element ″backingMapPluginCollections″ ein einziges Element ″backingMapPluginCollection″ mit der ID `customerPlugins`.

- Anzahl der Vorkommen: 0 bis 1
- Untergeordnetes Element: backingMapPluginCollection

Element ″backingMapPluginCollection″

Das Element ″backingMapPluginCollection″ definiert die BackingMap-Plug-ins und wird über das Attribut **id** identifiziert. Geben Sie das Attribut ″pluginCollectionRef″ an, um die Plug-ins zu referenzieren. Wenn Sie mehrere BackingMap-Plug-ins auf ähnliche Weise konfigurieren, kann jede BackingMap dasselbe Element ″backingMapPluginCollection″ referenzieren.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: bean

Attribute

id Identifiziert die backingMapPluginCollection und wird vom Attribut ″pluginCollectionRef″ des Elements ″backingMap″ referenziert. Jede ID muss eindeutig sein. Wenn der Wert des Attributs ″pluginCollectionRef″ nicht mit der ID eines der backingMapPluginCollection-Elemente übereinstimmt, schlägt die XML-Validierung fehl. Es können beliebig viele backingMap-Elemente ein einziges Element ″backingMapPluginCollection″ referenzieren. (Erforderlich)

```
<backingMapPluginCollection
(1)   id="id"
/>
```

Im folgenden Beispiel wird die Datei `companyGridCollection.xml` verwendet, um zu veranschaulichen, wie das Element ″backingMapPluginCollection″ verwendet wird. In dieser Datei verwendet die BackingMap ″Customer″ die backingMapPluginCollection ″customerPlugins″, um die BackingMap ″Customer″ mit einem LRUEvictor-Plug-in zu konfigurieren. Die BackingMaps ″Item″ und ″OrderLine″ referenzieren die backingMapPluginCollection ″collection2″. Für diese BackingMaps ist jeweils ein LFUEvictor-Plug-in definiert.

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

 <objectGrids>
<objectGrid name="CompanyGrid">
<backingMap name="Customer"
pluginCollectionRef="customerPlugins" />
   <backingMap name="Item" pluginCollectionRef="collection2"/>
   <backingMap name="OrderLine"
    pluginCollectionRef="collection2"/>
<backingMap name="Order"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
<backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor/>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="collection2">
   <bean id="Evictor"
     className="com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor/>
   <bean id="OptimisticCallback"
     className="com.ibm.websphere.samples.objectgrid.EmployeeOptimisticCallBackImpl"/>
  </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridCollection.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();

ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
BackingMap customerMap = companyGrid .defineMap("Customer");
LRUEvictor customerEvictor = new LRUEvictor();
customerMap.setEvictor(customerEvictor);

BackingMap itemMap = companyGrid.defineMap("Item");
LFUEvictor itemEvictor = new LFUEvictor();
itemMap.setEvictor(itemEvictor);

BackingMap orderLineMap = companyGrid.defineMap("OrderLine");
LFUEvictor orderLineEvictor = new LFUEvictor();
orderLineMap.setEvictor(orderLineEvictor);

BackingMap orderMap = companyGrid.defineMap("Order");
```

Element "querySchema"

Das Element "querySchema" definiert Beziehungen zwischen BackingMaps und identifiziert den Objekttyp jeder Map. Diese Informationen werden von ObjectQuery verwendet, um Zeichenfolgen in der Abfragesprache in Map-Zugriffsaufrufe zu übersetzen. Weitere Informationen finden Sie in den Details zur Definition eines ObjectQuery-Schemas im *Programmierhandbuch*.

- Anzahl der Vorkommen: 0 bis 1
- Untergeordnete Elemente: mapSchemas, relationships

Element "mapSchemas"

Jedes Element "querySchema" hat ein einziges Element "mapSchemas", das ein oder mehrere mapSchema-Elemente enthält.

- Anzahl der Vorkommen: 1
- Untergeordnetes Element: mapSchema

Element "mapSchema"

Ein Element "mapSchema" definiert den Typ der Objekte, die in einer BackingMap gespeichert werden, und enthält Anweisungen zum Zugriff auf die Daten.

- Anzahl der Vorkommen: 1 oder mehr
- Untergeordnetes Element: Ohne

Attribute

mapName
Gibt den Namen der BackingMap an, die dem Schema hinzugefügt werden soll. (Erforderlich)

valueClass
Gibt den Typ des Objekts an, der im Wertabschnitt der BackingMap gespeichert wird. (Erforderlich)

primaryKeyField
Gibt den Namen des Primärschlüsselattributs im Attribut "valueClass" an. Der Primärschlüssel muss auch im Schlüsselabschnitt der BackingMap gespeichert werden. (Optional)

accessType
Gibt an, wie die Abfragesteuerkomponente die persistenten Daten in den

valueClass-Objektinstanzen überwacht und auf diese zugreift. Wenn Sie das Attribut auf den Wert FIELD setzen, werden die Klassenfelder überwacht und dem Schema hinzugefügt. Wenn Sie das Attribut auf den Wert PROPERTY setzen, werden die Attribute, die den get- und is-Methoden zugeordnet sind, verwendet. Der Standardwert ist PROPERTY. (Optional)

```
<mapSchema
(1)   mapName="backingMapName"
(2)   valueClass="com.mycompany.OrderBean"
(3)   primaryKeyField="orderId"
(4)   accessType="PROPERTY" | "FIELD"
/>
```

Im folgenden Beispiel wird die Datei companyGridQuerySchemaAttr.xml verwendet, um eine Beispielkonfiguration für mapSchema zu veranschaulichen:

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

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

   <querySchema>
    <mapSchemas>
     <mapSchema mapName="Order"
      valueClass="com.mycompany.OrderBean"
      primaryKeyField="orderNumber"
      accessType="FIELD"/>
     <mapSchema mapName="Customer"
      valueClass="com.mycompany.CustomerBean"
      primaryKeyField="id"
      accessType="FIELD"/>
    </mapSchemas>
   </querySchema>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei companyGridQuerySchemaAttr.xml aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Schema definieren
QueryConfig queryCfg = new QueryConfig();
queryCfg.addQueryMapping(new QueryMapping(
    "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryMapping(new QueryMapping(
    "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
companyGrid.setQueryConfig(queryCfg);
```

Element "relationships"

Jedes Element "querySchema" hat kein oder ein Element "relationships", das eine oder mehrere Elemente "relationship" enthält.

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: relationship

Element "relationship"

Ein Element "relationship" definiert die Beziehung zwischen zwei BackingMaps und die Attribute im Attribut "valueClass", die für die Beziehungsbindung verwendet werden.

- Anzahl der Vorkommen: 1 oder mehr
- Untergeordnetes Element: Ohne

Attribute

source
Gibt den Namen der valueClass der Quellenseite einer Beziehung an. (Erforderlich)

target
Gibt den Namen der valueClass der Zielseite einer Beziehung an. (Erforderlich)

relationField
Gibt den Namen des Attributs in der Quellen-valueClass an, die auf das Ziel verweist. (Erforderlich)

invRelationField
Gibt den Namen des Attributs in der Ziel-valueClass an, die auf die Quelle verweist. Wenn Sie dieses Attribut nicht angeben, ist die Beziehung unidirektional. (Optional)

```
<mapSchema
(1)   source="com.mycompany.OrderBean"
(2)   target="com.mycompany.CustomerBean"
(3)   relationField="customer"
(4)   invRelationField="orders"
/>
```

Im folgenden Beispiel wird die Datei `companyGridQuerySchemaWithRelationshipAttr.xml` verwendet, um eine mapSchema-Musterkonfiguration zu veranschaulichen, die eine bidirektionale Beziehung enthält.

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

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

   <querySchema>
    <mapSchemas>
     <mapSchema mapName="Order"
      valueClass="com.mycompany.OrderBean"
      primaryKeyField="orderNumber"
      accessType="FIELD"/>
     <mapSchema mapName="Customer"
      valueClass="com.mycompany.CustomerBean"
      primaryKeyField="id"
      accessType="FIELD"/>
    </mapSchemas>
    <relationships>
     <relationship
      source="com.mycompany.OrderBean"
      target="com.mycompany.CustomerBean"
      relationField="customer"/>
      invRelationField="orders"/>
    </relationships>
   </querySchema>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Der folgende Mustercode veranschaulicht den programmgesteuerten Ansatz, um dieselbe Konfiguration wie mit der Datei `companyGridQuerySchemaWithRelationshipAttr.xml` aus dem vorherigen Beispiel zu erhalten.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
```

```
companyGrid.defineMap("Customer");

// Schema definieren
QueryConfig queryCfg = new QueryConfig();
queryCfg.addQueryMapping(new QueryMapping(
    "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryMapping(new QueryMapping(
    "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryRelationship(new QueryRelationship(
     OrderBean.class.getName(), CustomerBean.class.getName(), "customer", "orders"));
companyGrid.setQueryConfig(queryCfg);
```

Element ″timeBasedDBUpdate″

Ein Element ″timeBasedDBUpdate″ definiert eine Konfiguration für eine zeitbasierte Datenbankaktualisierungskomponente. Ein Element ″timeBasedDBUpdate″ enthält Informationen dazu, wie oft neu eingefügte oder aktualisierte Datensätze aus der Datenbank mit Java Persistence API (JPA) abgerufen werden und wie die Daten in den entsprechenden ObjectGrid-Maps aktualisiert werden.

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: Ohne

Attribute

entityClass
Gibt den Namen der Entitätsklasse an, die für die Interaktion mit dem JPA-Provider verwendet wird. Der Entitätsklassenname wird verwendet, um JPA-Entitäten über Entitätsabfragen abzurufen. (Erforderlich)

persistenceUnitName
Gibt den Namen der JPA-Persistenzeinheit für die Erstellung einer JPA-Entitätsmanager-Factory an. Der Standardwert ist der Name der ersten in der Datei `persistence.xml` definierten Persistenzeinheit. (Optional)

mode
Gibt den Modus für die zeitbasierte Datenbankaktualisierung an. Der Standardmodus für die zeitbasierte Datenbankaktualisierung ist INVALIDATE_ONLY. Der Modus INVALIDATE_ONLY gibt an, dass die Einträge in der ObjectGrid-Map ungültig gemacht, wenn die entsprechenden Datensätze in der Datenbank geändert wurden. Der Modus UPDATE_ONLY gibt an, dass die vorhandenen Einträge in der ObjectGrid-Map mit den aktuellen Werten aus der Datenbank aktualisiert werden. Alle neu in die Datenbank eingefügten Datensätze werden jedoch ignoriert. Der Modus INSERT_UPDATE gibt an, dass die vorhandenen Einträge in der ObjectGrid-Map mit den aktuellen Werten aus der Datenbank aktualisiert werden. Außerdem werden alle neu in die Datenbank eingefügten Datensätze in die ObjectGrid-Map eingefügt. (Optional)

timestampField
Gibt den Namen des Zeitmarkenfelds an. Ein Zeitmarkenfeldwert wird verwendet, um die Zeit oder Folge der letzten Aktualisierung eines Datensatzes im Datenbank-Back-End anzugeben. (Optional)

jpaPropertyFactory
Gibt den Namen der JPAPropertyFactory-Implementierungsklasse bzw. Spring-Bean an. Die Schnittstelle ″com.ibm.websphere.objectgrid.jpa-.JPAPropertyFactory″ wird verwendet, um eine Persistenzeigenschaftentabelle zu integrieren, die die JPA-Standardeigenschaften überschreibt. Verwenden Sie Beans des Spring-Frameworks, wenn zusätzliche Attribute in der JPAPropertyFactory-Instanz gesetzt werden müssen. Weitere Informationen finden Sie im Abschnitt „Integration mit dem Spring-Framework” auf Seite 210. (Optional)

```
<timeBasedDBUpdate
(1)   persistenceUnitName="SamplePU"
(2)   mode="INVALIDATE_ONLY" | "UPDATE_ONLY" | "INSERT_UPDATE"
(3)   timestampField="TIMESTAMP"
(4)   entityClass="entity class"
(5)   jpaPropertyFactory="JPA property factory class" | "{spring}bean name"
/>
```

Element "streamQuerySet"

Das Element "streamQuerySet" ist das Ausgangselement für die Definition eines Abfragesatzes für Datenströme.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnete Elemente: stream, view

Element "stream"

Das Element "stream" stellt einen Datenstrom für die Abfragesteuerkomponente für Datenströme dar. Jedes Attribut des Elements "stream" entspricht einer Methode in der Schnittstelle "StreamMetadata".

- Anzahl der Vorkommen: 1 bis viele
- Untergeordnetes Element: basic

Attribute

name
Gibt den Namen des Datenstroms an. Die Validierung schlägt fehl, wenn dieses Attribut nicht angegeben wird. (Erforderlich)

valueClass
Gibt den Klassentyp des Werts an, der in der ObjectMap des Datenstroms gespeichert wird. Der Klassentyp wird verwendet, um das Objekt in Datenstromereignisse zu konvertieren und um eine SQL-Anweisung zu generieren, wenn die Anweisung nicht angegeben ist. (Erforderlich)

sql
Gibt die SQL-Anweisung des Datenstroms an. Wenn Sie diese Eigenschaft nicht angeben, wird eine Datenstrom-SQL durch Reflexion der Attribute bzw. Zugriffsmethoden im Attribut "valueClass" bzw. durch Verwendung der Tupelattribute der Entitätsmetadaten generiert. (Optional)

access
Gibt den Typ für den Zugriff auf die Attribute der Wertklasse an. Wenn Sie das Attribut auf den Wert `FIELD` setzen, werden die Attribute durch Java-Reflexion direkt aus den Feldern abgerufen. Andernfalls werden Zugriffsmethoden für das Lesen der Attribute verwendet. Der Standardwert ist `PROPERTY`. (Optional)

```
<stream
(1)   name="streamName"
(2)   valueClass="streamMapClassType"
(3)   sql="streamSQL create stream stockQuote
       keyed by t ( transactionvolume INTEGER, price DECIMAL (9,2),
        issue VARCHAR(100) );"
(4)   access="PROPERTY" | "FIELD"
/>
```

Element "view"

Das Element "view" stellt eine Datenstromabfragesicht dar. Jedes Element "stream" entspricht einer Methode in der Schnittstelle "ViewMetadata".

- Anzahl der Vorkommen: 1 bis viele
- Untergeordnete Elemente: basic, id

Attribute

name
Gibt den Namen der Sicht an. Die Validierung schlägt fehl, wenn dieses Attribut nicht angegeben wird. (Erforderlich)

sql
Gibt die SQL des Datenstroms an, die die Sichtkonvertierung definiert. Die Validierung schlägt fehl, wenn dieses Attribut nicht angegeben wird. (Erforderlich)

valueClass
Gibt den Klassentyp des Werts an, der in dieser Sicht der ObjectMap gespeichert wird. Der Klassentyp wird verwendet, um die Sichtereignisse in das richtige Tupelformat zu konvertieren, das mit diesem Klassentyp kompatibel ist. Wenn Sie den Klassentyp nicht angeben, wird ein Standardformat gemäß den Spaltendefinition in Stream Processing Technology Structured Query Language (SPTSQL) verwendet. Wenn Entitätsmetadaten für diese Sicht-Map definiert sind, darf dieses Attribut nicht verwendet werden. Stattdessen werden die Entitätsmetadaten verwendet. (Optional)

access
Gibt den Typ für den Zugriff auf die Attribute der Wertklasse an. Wenn Sie den Zugriffstyp auf `FIELD` setzen, werden die Spaltenwerte durch Java-Reflexion direkt aus den Feldern abgerufen. Andernfalls werden Zugriffsmethoden für das Definieren der Attribute verwendet. Der Standardwert ist `PROPERTY`. (Optional)

```
<view
(1)   name="viewName"
(2)   valueClass="viewMapValueClass"
(3)   sql="viewSQL CREATE VIEW last5MinuteAvgPrice AS
      SELECT issue, avg(price) as totalVolume
        FROM (SELECT * FROM stockQuote FETCH LATEST 5 MINUTES) group by issue;"/>
(4)   access="PROPERTY" | "FIELD"
/>
```

Element ″basic″

Das Element ″basic″ wird verwendet, um eine Zuordnung des Attributnamens in der Wertklasse bzw. den Entitätsmetadaten zu der in SPTSQL definierten Spalte zu definieren.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

```
<basic
(1)   name="attributeName"
(2)   column="columnName"
/>
```

Element ″id″

Das Element ″id″ wird für die Zuordnung eines Schlüsselattributs verwendet.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

```
<id
(1)   name="idName"
(2)   column="columnName"
/>
```

Im folgenden Beispiel wird die Datei `StreamQueryApp2.xml` verwendet, um zu veranschaulichen, wie die Attribut eines Datenstromabfragesatzes konfiguriert werden. Der Datenstromabfragesatz "_stockQuoteSQS_" hat einen Datenstrom und eine Sicht. Datenstrom und Sicht definieren einen Namen, die Werteklasse, die SQL und den Zugriffstyp. Der Datenstrom definiert auch ein Element "basic", das angibt, dass das Attribut "volume" in der Klasse "StockQuote" der in der SQL-Anweisung definierten SQL-Spalte "transactionvolume" zugeordnet ist.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
 <objectGrids>
  <objectGrid name="og1">
   <backingMap name="stockQuote" readOnly="false" copyKey="true"
    streamRef="stockQuote"/>
   <backingMap name="last5MinuteAvgPrice" readOnly="false" copyKey="false"
    viewRef="last5MinuteAvgPrice"/>

   <streamQuerySet name="stockQuoteSQS">
    <stream
     name="stockQuote"
     valueClass="com.ibm.ws.objectgrid.streamquery.sample.guide.StockQuote"
     sql="create stream stockQuote
         keyed by t ( transactionvolume INTEGER, price DECIMAL (9,2),
        issue VARCHAR(100) );"
     access="FIELD">
     <basic name="volume" column="transactionvolume"/>
    </stream>

    <view
     name="last5MinuteAvgPrice"
     valueClass="com.ibm.ws.objectgrid.streamquery.sample.guide.AveragePrice"
     sql="CREATE VIEW last5MinuteAvgPrice AS SELECT issue, avg(price) as avgPrice
       FROM (SELECT * FROM stockQuote FETCH LATEST 5 MINUTES) group by issue;"
     access="FIELD"
    </view>
   </streamQuerySet>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Datei `emd.xsd`

Verwenden Sie die XML-Schemadefinition für Entitätsmetadaten, um eine XML-Deskriptordatei zu erstellen und ein Entitätsschema für WebSphere eXtreme Scale zu definieren.

Beschreibungen der einzelnen Elemente und Attribute in der Datei `emd.xsd` finden Sie im „XML-Deskriptordatei für Entitätsmetadaten" auf Seite 178.

Datei `emd.xsd`

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:emd="http://ibm.com/ws/projector/config/emd"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://ibm.com/ws/projector/config/emd"
 elementFormDefault="qualified" attributeFormDefault="unqualified"
 version="1.0">

 <xsd:element name="entity-mappings"
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="description" type="xsd:string" minOccurs="0"/>
    <xsd:element name="entity" type="emd:entity" minOccurs="1" maxOccurs="unbounded"/>
   </xsd:sequence>
  </xsd:complexType>
  <xsd:unique name="uniqueEntityClassName">
```

```
  <xsd:selector xpath="emd.entity"/>
  <xsd:field xpath="@class-name"/>
 </xsd:unique>
</xsd:element>

<xsd:complexType name="entity">
 <xsd:sequence>
  <xsd:element name="description" type="xsd:string" minOccurs="0"/>
  <xsd:element name="id-class" type="emd:id-class" minOccurs="0"/>
  <xsd:element name="attributes" type="emd:attributes" minOccurs="0"/>
  <xsd:element name="entity-listeners" type="emd:entity-listeners" minOccurs="0"/>
  <xsd:element name="pre-persist" type="emd:pre-persist" minOccurs="0"/>
  <xsd:element name="post-persist" type="emd:post-persist" minOccurs="0"/>
  <xsd:element name="pre-remove" type="emd:pre-remove" minOccurs="0"/>
  <xsd:element name="post-remove" type="emd:post-remove" minOccurs="0"/>
  <xsd:element name="pre-invalidate" type="emd:pre-invalidate" minOccurs="0"/>
  <xsd:element name="post-invalidate" type="emd:post-invalidate" minOccurs="0"/>
  <xsd:element name="pre-update" type="emd:pre-update" minOccurs="0"/>
  <xsd:element name="post-update" type="emd:post-update" minOccurs="0"/>
  <xsd:element name="post-load" type="emd:post-load" minOccurs="0"/>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="class-name" type="xsd:string" use="required"/>
 <xsd:attribute name="access" type="emd:access-type"/>
 <xsd:attribute name="schemaRoot" type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="attributes">
 <xsd:sequence>
  <xsd:choice>
   <xsd:element name="id" type="emd:id" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:choice>
  <xsd:element name="basic" type="emd:basic" minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="version" type="emd:version" minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="many-to-one" type="emd:many-to-one" minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="one-to-many" type="emd:one-to-many" minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="one-to-one" type="emd:one-to-one" minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="many-to-many" type="emd:many-to-many" minOccurs="0" maxOccurs="unbounded"/>
  <xsd:element name="transient" type="emd:transient" minOccurs="0" maxOccurs="unbounded"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:simpleType name="access-type">
 <xsd:restriction base="xsd:token">
  <xsd:enumeration value="PROPERTY"/>
  <xsd:enumeration value="FIELD"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="id-class">
 <xsd:attribute name="class-name"> type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="id">
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="alias" type="xsd:string" use="optional"/>
</xsd:complexType>

<xsd:complexType name="transient">
 <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="basic">
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="alias" type="xsd:string"/>
 <xsd:attribute name="fetch" type="emd:fetch-type"/>
</xsd:complexType>

<xsd:simpleType name="fetch-type">
 <xsd:restriction base="xsd:token">
  <xsd:enumeration value="LAZY"/>
  <xsd:enumeration value="EAGER"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:complexType name="many-to-one">
 <xsd:sequence>
  <xsd:element name="cascade" type="emd:cascade-type" minOccurs="0"/>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="alias" type="xsd:string"/>
 <xsd:attribute name="target-entity" type="xsd:string"/>
 <xsd:attribute name="fetch" type="emd:fetch-type"/>
 <xsd:attribute name="id" type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="one-to-one">
 <xsd:sequence>
  <xsd:element name="cascade" type="emd:cascade-type" minOccurs="0"/>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
```

```
 <xsd:attribute name="alias" type="xsd:string"/>
 <xsd:attribute name="target-entity" type="xsd:string"/>
 <xsd:attribute name="fetch" type="emd:fetch-type"/>
 <xsd:attribute name="mapped-by" type="xsd:string"/>
 <xsd:attribute name="id" type="xsd:boolean"/>
</xsd:complexType>

<xsd:complexType name="one-to-many">
 <xsd:sequence>
  <xsd:element name="order-by" type="emd:order-by" minOccurs="0"/>
  <xsd:element name="cascade" type="emd:cascade-type" minOccurs="0"/>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="alias" type="xsd:string"/>
 <xsd:attribute name="target-entity" type="xsd:string"/>
 <xsd:attribute name="fetch" type="emd:fetch-type"/>
 <xsd:attribute name="mapped-by" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="many-to-many">
 <xsd:sequence>
  <xsd:element name="order-by" type="emd:order-by" minOccurs="0"/>
  <xsd:element name="cascade" type="emd:cascade-type" minOccurs="0"/>
 </xsd:sequence>
 <xsd:attribute name="name" type="xsd:string" use="required"/>
 <xsd:attribute name="alias" type="xsd:string"/>
 <xsd:attribute name="target-entity" type="xsd:string"/>
 <xsd:attribute name="fetch" type="emd:fetch-type"/>
 <xsd:attribute name="mapped-by" type="xsd:string"/>
</xsd:complexType>

<xsd:simpleType name="order-by">
 <xsd:restriction base="xsd:string"/>
</xsd:simpleType>

<xsd:complexType name="cascade-type">
 <xsd:sequence>
  <xsd:element name="cascade-all" type="emd:emptyType" minOccurs="0"/>
  <xsd:element name="cascade-persist" type="emd:emptyType" minOccurs="0"/>
  <xsd:element name="cascade-remove" type="emd:emptyType" minOccurs="0"/>
  <xsd:element name="cascade-invalidate" type="emd:emptyType" minOccurs="0"/>
  <xsd:element name="cascade-merge" type="emd:emptyType" minOccurs="0"/>
  <xsd:element name="cascade-refresh" type="emd:emptyType" minOccurs="0"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="emptyType/>

<xsd:complexType name="version">
 <xsd:attribute name="name" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="entity-listeners">
 <xsd:sequence>
  <xsd:element name="entity-listener" type="emd:entity-listener" minOccurs="0" maxOccurs="unbounded"/>
 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="entity-listener">
 <xsd:sequence>
  <xsd:element name="pre-persist" type="emd:pre-persist" minOccurs="0"/>
  <xsd:element name="post-persist" type="emd:post-persist" minOccurs="0"/>
  <xsd:element name="pre-remove" type="emd:pre-remove" minOccurs="0"/>
  <xsd:element name="post-remove" type="emd:post-remove" minOccurs="0"/>
  <xsd:element name="pre-invalidate" type="emd:pre-invalidate" minOccurs="0"/>
  <xsd:element name="post-invalidate" type="emd:post-invalidate" minOccurs="0"/>
  <xsd:element name="pre-update" type="emd:pre-update" minOccurs="0"/>
  <xsd:element name="post-update" type="emd:post-update" minOccurs="0"/>
  <xsd:element name="post-load" type="emd:post-load" minOccurs="0"/>
 </xsd:sequence>
 <xsd:attribute name="class-name" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="pre-persist">
 <xsd:attribute name="method-name" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="post-persist">
 <xsd:attribute name="method-name" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="pre-remove">
 <xsd:attribute name="method-name" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="post-remove">
 <xsd:attribute name="method-name" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="pre-invalidate">
 <xsd:attribute name="method-name" type="xsd:string" use="required"/>
```

```
 </xsd:complexType>

 <xsd:complexType name="post-invalidate">
  <xsd:attribute name="method-name" type="xsd:string" use="required"/>
 </xsd:complexType>

 <xsd:complexType name="pre-update">
  <xsd:attribute name="method-name" type="xsd:string" use="required"/>
 </xsd:complexType>

 <xsd:complexType name="post-update">
  <xsd:attribute name="method-name" type="xsd:string" use="required"/>
 </xsd:complexType>

 <xsd:complexType name="post-load">
  <xsd:attribute name="method-name" type="xsd:string" use="required"/>
 </xsd:complexType>

</xsd:schema>
```

XML-Sicherheitsdeskriptordatei

Verwenden Sie eine ObjectGrid-XML-Sicherheitsdeskriptordatei, um eine Implementierungstopologie von eXtreme Scale mit aktivierter Sicherheit zu konfigurieren. In diesem Abschnitt werden verschiedene Konfigurationen anhand von XML-Musterdateien veranschaulicht.

Jedes Element und Attribut der XML-Datei wird in der folgenden Liste beschrieben. Verwenden Sie die Beispiele, um sich mit der Verwendung dieser Elemente und Attribute für die Konfiguration der Umgebung vertraut zu machen.

Element "securityConfig"

Das Element "securityConfig" ist das Ausgangselement der ObjectGrid-XML-Sicherheitsdatei. Dieses Element konfiguriert den Namespace der Datei und die Schemaposition. Das Schema wird in der Datei `objectGridSecurity.xsd` definiert.

- Anzahl der Vorkommen: 1
- Untergeordnete Elemente: security

Element "security"

Verwenden Sie das Element "security", um die Sicherheit eines ObjectGrids zu definieren.

- Anzahl der Vorkommen: 1
- Untergeordnete Elemente: authenticator, adminAuthorization und systemCredentialGenerator

Attribute

securityEnabled
Aktiviert die Sicherheit für das Grid, wenn es den Wert "true" hat. Der Standardwert ist "false". Wenn das Attribut den Wert "false" hat, ist die Grid-weite Sicherheit inaktiviert. Weitere Informationen finden Sie im Abschnitt „Grid-Sicherheit" auf Seite 310. (Optional)

singleSignOnEnabled
Wenn Sie das Attribut auf "true" setzen, kann ein Client eine Verbindung zu jedem Server herstellen, nachdem er bei einem der Server authentifiziert wurde. Hat das Attribut den Wert "false", muss ein Client sich bei jedem Server authentifizieren, bevor er eine Verbindung herstellen kann. Der Standardwert ist "false". (Optional)

loginSessionExpirationTime
Gibt die Verfallszeit der Anmeldesitzung in Sekunden an. Wenn die Anmeldesitzung abläuft, muss sich der Client erneut authentifizieren. (Optional)

adminAuthorizationEnabled
Aktiviert die Verwaltungsberechtigung. Wenn dieses Attribut den Wert ″true″ hat, müssen alle Verwaltungs-Tasks berechtigt werden. Der verwendete Berechtigungsmechanismus wird mit dem Wert des Attributs ″adminAuthorizationMechanism″ angegeben. Der Standardwert ist ″false″. (Optional)

adminAuthorizationMechanism
Gibt an, welcher Berechtigungsmechanismus zu verwenden ist. WebSphere eXtreme Scale unterstützt zwei Berechtigungsmechanismen: Java Authentication and Authorization Service (JAAS) und angepasste Berechtigung. Der Berechtigungsmechanismus JAAS verwendet den richtlinienbasierten JAAS-Standardansatz. Wenn Sie JAAS als Berechtigungsmechanismus festlegen möchten, setzen Sie das Attribut auf AUTHORIZATION_MECHANISM_JAAS. Der angepasste Berechtigungsmechanismus verwendet eine vom Benutzer integrierte AdminAuthorization-Implementierung. Wenn Sie einen angepassten Berechtigungsmechanismus angeben möchten, setzen Sie das Attribut auf AUTHORIZATION_MECHANISM_CUSTOM. Weitere Informationen zur Verwendung dieser beiden Mechanismen finden Sie im Abschnitt „Anwendungsclientberechtigung” auf Seite 314. (Optional)

Die folgende Datei `security.xml` ist eine Musterkonfiguration zum Aktivieren der Sicherheit eines eXtreme-Scale-Grids:

```
security.xml

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

    <security securityEnabled="true" singleSignOnEnabled="true"
        loginSessionExpirationTime="20"
        adminAuthorizationEnabled="true"
        adminAuthorizationMechanism="AUTHORIZATION_MECHANISM_JAAS" >

        <authenticator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator">
        </authenticator>

        <systemCredentialGenerator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator">
            <property name="properties" type="java.lang.String" value="runAs" description="Using runAs subject" />
        </systemCredentialGenerator>

    </security></securityConfig>
```

Element ″authenticator″

Authentifiziert Clients bei eXtreme-Scale-Servern im Grid. Die Klasse, die mit dem Attribut ″className″ angegeben wird, muss die Schnittstelle ″com.ibm.websphere.objectgrid.security.plugins.Authenticator″ implementieren. Der Authentifikator kann Eigenschaften verwenden, um Methoden in der Klasse aufzurufen, die mit dem Attribut ″className″ angegeben wird. Weitere Informationen zur Verwendung von Eigenschaften finden Sie in der Beschreibung des Elements ″property″.

In der vorherigen Beispieldatei `security.xml` wurde die Klasse ″com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator″ als Authentifikator angegeben. Diese Klasse implementiert die Schnittstelle ″com.ibm.websphere.objectgrid.security.plugins.Authenticator″.

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: property

Attribute

className
Gibt eine Klasse an, die die Schnittstelle ″com.ibm.websphere.objectgrid.security.plugins.Authenticator″ implementiert. Verwenden Sie diese Klasse, um Clients bei Servern im eXtreme-Scale_Grid zu authentifizieren. (Erforderlich)

Element ″adminAuthorization″

Verwenden Sie das Element ″adminAuthorization″, um den Verwaltungszugriff auf das Grid zu konfigurieren.

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: property

Attribute

className
Gibt eine Klasse an, die die Schnittstelle ″com.ibm.websphere.objectgrid.security.plugins.AdminAuthorization″ implementiert. (Erforderlich)

Element ″systemCredentialGenerator″

Verwenden Sie ein Element ″systemCredentialGenerator″, um einen Generator für Systemberechtigungsnachweise zu konfigurieren. Dieses Element gilt nur für eine dynamische Umgebung. Im dynamischen Konfigurationsmodell stellt der dynamische Containerserver als eXtreme-Scale-Client eine Verbindung zum Katalogserver her, und der Katalogserver kann ebenfalls als Client eine Verbindung zum eXtreme-Scale-Containerserver herstellen. Dieser Generator für Systemberechtigungsnachweise wird für die Darstellung einer Factory für die Systemberechtigungsnachweise verwendet.

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: property

Attribute

className
Gibt eine Klasse an, die die Schnittstelle ″com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator″ implementiert. (Erforderlich)

Ein Beispiel zur Verwendung des Elements ″systemCredentialGenerator″ entnehmen Sie der vorherigen Datei `security.xml`. In diesem Beispiel ist com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenCredentialGenerator der Generator für Systemberechtigungsnachweise, der das RunAs-Subject-Objekt aus dem Thread abruft.

Element ″property″

Ruft set-Methoden in den Klassen ″authenticator″ und ″adminAuthorization″ auf. Der Name der Eigenschaft entspricht einer set-Methode im Attribut ″className″ des Elements ″authenticator″ bzw. ″adminAuthorization″.

- Anzahl der Vorkommen: 0 oder mehr
- Untergeordnetes Element: property

Attribute

name
Der Name der Eigenschaft. Der Wert, der diesem Attribut zugeordnet wird, muss einer set-Methode in der Klasse entsprechen, die mit dem Attribut ″class-

Name″ der übergeordneten Bean angegeben wird. Wenn das Attribut ″class-Name″ der Bean beispielsweise auf ″com.ibm.MyPlugin″ gesetzt ist und der Name der angegebenen Eigenschaft ″size″ lautet, muss die Klasse ″com.ibm-.MyPlugin″ eine Methode ″setSize″ haben. (Erforderlich)

type
Gibt den Typ der Eigenschaft an. Der Typ des Parameters, der an die mit dem Attribut ″name″ angegebene set-Methode übergeben wird. Die gültigen Werte sind primitive Java-Typen, ihre java.lang-Gegenstücke und java.lang.String. Die Attribute ″name″ und ″type″ müssen einer Methodensignatur im Attribut ″className″ der Bean entsprechen. Wenn der Name beispielsweise ″size″ und der Typ ″int″ ist, muss eine Methode ″setSize(int)″ in der Klasse vorhanden sein, die mit dem Attribut ″className″ für die Bean angegeben wurde. (Erforderlich)

value
Gibt den Wert der Eigenschaft an. Dieser Wert wird in den mit dem Attribut ″type″ angegebenen Typ konvertiert und anschließend als Parameter in dem Aufruf der set-Methode verwendet, die mit den Attributen ″name″ und ″type″ angegeben wurde. Der Wert dieses Attributs wird nicht validiert. Der Plug-in-Implementierer muss sicherstellen, dass der übergebene Wert gültig ist. (Erforderlich)

description
Gibt eine Beschreibung der Eigenschaft an. (Optional)

Weitere Informationen finden Sie im Abschnitt „Datei `objectGridSecurity.xsd`”.

Datei `objectGridSecurity.xsd`

Verwenden Sie das folgende ObjectGrid-XML-Sicherheitsschema, um die Sicherheit für eine eXtreme-Scale-Implementierung zu aktivieren.

Im Abschnitt „XML-Sicherheitsdeskriptordatei” auf Seite 192 finden Sie Beschreibungen der Elemente und Attribute, die in der Datei `objectGridSecurity.xsd` definiert sind.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:cc="http://ibm.com/ws/objectgrid/config/security"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://ibm.com/ws/objectgrid/config/security"
  elementFormDefault="qualified">

 <xsd:element name="securityConfig">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="security" type="cc:security" />
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>

 <xsd:complexType name="security">
  <xsd:sequence>
   <xsd:element name="authenticator" type="cc:bean" minOccurs="0"
    maxOccurs="1" />
   <xsd:element name="adminAuthorization" type="cc:bean" minOccurs="0"
    maxOccurs="1" />
   <xsd:element name="systemCredentialGenerator" type="cc:bean" minOccurs="0"
    maxOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="securityEnabled" type="xsd:boolean" use="optional" />
  <xsd:attribute name="singleSignOnEnabled" type="xsd:boolean" use="optional"/>
  <xsd:attribute name="loginSessionExpirationTime" type="xsd:int" use="optional"/>
  <xsd:attribute name="adminAuthorizationMechanism" type="cc:adminAuthorizationMechanism"
   use="optional"/>
  <xsd:attribute name="adminAuthorizationEnabled" type="xsd:boolean" use="optional" />
 </xsd:complexType>

<xsd:complexType name="bean">
```

```
  <xsd:sequence>
   <xsd:element name="property" type="cc:property" maxOccurs="unbounded" minOccurs="0" />
  </xsd:sequence>
  <xsd:attribute name="className" type="xsd:string" use="required"/>
 </xsd:complexType>

<xsd:complexType name="property">
  <xsd:attribute name="name" type="xsd:string" use="required" />
<xsd:attribute name="value" type="xsd:string" use="required" />
  <xsd:attribute name="type" type="cc:propertyType" use="required" />
  <xsd:attribute name="description" type="xsd:string" use="optional" />
 </xsd:complexType>

<xsd:simpleType name="propertyType">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="java.lang.Boolean"/>
   <xsd:enumeration value="boolean"/>
   <xsd:enumeration value="java.lang.String"/>
   <xsd:enumeration value="java.lang.Integer"/>
   <xsd:enumeration value="int"/>
   <xsd:enumeration value="java.lang.Double"/>
   <xsd:enumeration value="double"/>
   <xsd:enumeration value="java.lang.Byte"/>
   <xsd:enumeration value="byte"/>
   <xsd:enumeration value="java.lang.Short"/>
   <xsd:enumeration value="short"/>
   <xsd:enumeration value="java.lang.Long"/>
   <xsd:enumeration value="long"/>
   <xsd:enumeration value="java.lang.Float"/>
   <xsd:enumeration value="float"/>
   <xsd:enumeration value="java.lang.Character"/>
   <xsd:enumeration value="char"/>
  </xsd:restriction>
 </xsd:simpleType>

 <xsd:simpleType name="adminAuthorizationMechanism">
  <xsd:restriction base="xsd:string">
   <xsd:enumeration value="AUTHORIZATION_MECHANISM_JAAS" />
   <xsd:enumeration value="AUTHORIZATION_MECHANISM_CUSTOM"/>
  </xsd:restriction>
 </xsd:simpleType>

</xsd:schema>
```

Referenz der Eigenschaftendatei

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

Mustereigenschaftendateien

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

- sampleServer.properties
- sampleClient.properties

Veraltete Systemeigenschaften

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

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

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

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

Zugehörige Konzepte

„Transport Layer Security und Secure Sockets Layer" auf Seite 317
WebSphere eXtreme Scale unterstützt TCP/IP und Transport Layer Security/Secure Sockets Layer (TLS/SSL) für die sichere Kommunikation zwischen Client und Server in dynamischen Implementierungsmodellen.

Zugehörige Tasks

„Eigenständige Server von WebSphere eXtreme Scale starten" auf Seite 228
Wenn Sie eine eigenständige Konfiguration von WebSphere eXtreme Scale verwenden, setzt sich die Umgebung aus Katalogservern, Containerservern und eXtreme-Scale-Clientprozessen zusammen. Sie müssen diese Prozesse manuell konfigurieren und starten.

„WebSphere eXtreme Scale mit WebSphere Application Server verwalten" auf Seite 242
Sie können Katalogservice- und Containerserverprozesse in WebSphere Application Server ausführen. Der Prozess zum Konfigurieren dieser Server unterscheidet sich von dem in einer eigenständigen Konfiguration. Der Katalogservice kann automatisch in Servern oder im Deployment Manager von WebSphere Application Server gestartet werden. Der Containerprozess wird gestartet, wenn eine eXtreme-Scale-Anwendung in der Umgebung von WebSphere Application Server implementiert und gestartet wird.

Servereigenschaftendatei

Die Servereigenschaftendatei enthält verschiedene Eigenschaften, mit denen die verschiedenen Einstellungen für Ihren Server definiert werden, z. B. Trace-Einstellungen, Protokollierung und Sicherheitskonfiguration. Die Servereigenschaftendatei wird vom Katalogservice und von den Containerservern verwendet.

Musterservereigenschaftendatei

Sie können die Datei `sampleServer.properties`, die im Verzeichnis *Stammverzeichnis_von_eXtreme_Scale*`/properties` enthalten ist, verwenden, um eine eigene Eigenschaftendatei zu erstellen.

Servereigenschaftendatei angeben

Sie können die Servereigenschaftendatei mit einer der folgenden Methoden angeben. Wenn Sie eine Einstellung über einen der Einträge weiter hinten in der Liste angeben, wird die vorherige Einstellung überschrieben. Geben Sie beispielsweise einen Systemeigenschaftswert für die Servereigenschaftendatei an, überschreiben die Eigenschaften in dieser Datei die Werte in der Datei `objectGridServer.properties`, die im Klassenpfad enthalten ist.

1. Sie können irgendeine korrekt benannte Datei angeben, die im Klassenpfad enthalten ist. Wenn Sie diese korrekt benannte Datei im Arbeitsverzeichnis ablegen, wird die Datei nicht gefunden, sofern das Arbeitsverzeichnis nicht im Klassenpfad enthalten ist. Der verwendete Name lautet:

   ```
   objectGridServer.properties
   ```

2. Sie können eine Datei aus dem Systemarbeitsverzeichnis als Systemeigenschaft in einer eigenständigen Konfiguration oder in einer Konfiguration von WebSphere Application Server angeben. Die Datei darf nicht im Klassenpfad enthalten sein:

   `-Dobjectgrid.server.props=`*`Dateiname`*

3. Sie können die Datei als Parameter angeben, wenn Sie den Befehl startOgServer ausführen. Sie können diese Eigenschaften manuell überschreiben, um eine Datei im Systemarbeitsverzeichnis anzugeben.

   `-serverProps` *`Dateiname`*

4. Sie können die Datei als Korrekturwert mit den Methoden ″ServerFactory.getServerProperties″ und ″ServerFactory.getCatalogServerProperties″ über das Programm angeben. Das Objekt wird mit Daten aus den Eigenschaftendateien gefüllt.

Servereigenschaften

Allgemeine Eigenschaften

workingDirectory
Gibt die Position an, an die die Ausgabe des Containerservers geschrieben wird. Wenn dieser Wert nicht angegeben wird, wird die Ausgabe in ein Verzeichnis `log` im aktuellen Verzeichnis geschrieben. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

Standardwert: Ohne

traceSpec
Aktiviert die Trace-Erstellung und die Trace-Spezifikationszeichenfolge für den Containerserver. Die Trace-Erstellung ist standardmäßig inaktiviert. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

Standardwert: `*=all=disabled`

traceFile
Gibt den Namen einer Datei an, in die Trace-Informationen geschrieben werden sollen. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

systemStreamToFileEnabled
Ermöglicht dem Server, die SystemOut-, SystemErr- und Trace-Ausgabe in eine Datei zu schreiben. Wenn Sie diese Eigenschaft auf `false` setzen, wird die Ausgabe nicht in eine Datei geschrieben, sondern in der Konsole ausgegeben.

Standardwert: `true`

enableMBeans
Aktiviert Managed Beans (MBean) des ObjectGrid-Containers. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

Standardwert: `true`

serverName
Legt den Servernamen für die Identifikation des Servers fest. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

zoneName
Legt den Namen der Zone fest, zu der der Server gehört. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

HAManagerPort
Gibt die Nummer des vom High Availability Manager verwendeten Ports

an. Wenn Sie diese Eigenschaft nicht definieren, generiert der Katalogservice automatisch einen verfügbaren Port. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

listenerHost
Gibt den Namen des Hosts an, zu dem der Object Request Broker (ORB) eine Bindung herstellen soll. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

listenerPort
Gibt die Nummer des Ports an, zu dem der Object Request Broker (ORB) eine Bindung herstellen soll. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

JMXServicePort
Gibt die Nummer des Ports an, an dem der MBean-Server empfangsbereit sein soll. Diese Eigenschaft gilt für den Containerserver und den Katalogservice.

Eigenschaften des Containerservers

statsSpec
Gibt die Statistikspezifikation für den Containerserver an.

Beispiel:

```
all=disabled
```

memoryThresholdPercentage
Legt den Speicherschwellenwert für die speicherbasierte Bereinigung fest. Dieser Wert gibt an, bei welchem Prozentsatz der maximalen Heap-Speicherbelegung in der Java Virtual Machine (JVM) die Bereinigung stattfindet. Der Standardwert ist `-1`, und gibt an, dass kein Speicherschwellenwert definiert ist. Wenn Sie die Eigenschaft ″memoryThresholdPercentage″ definieren, wird der MemoryPoolMXBean-Wert auf der Basis des bereitgestellten Werts gesetzt. Weitere Informationen finden Sie unter MemoryPoolMXBean interface in der Java-API-Spezifikation. Die Bereinigung findet jedoch nur statt, wenn sie in einem Bereinigungsprogramm aktiviert ist. Informationen zum Aktivieren der speicherbasierten Bereinigung finden Sie in der Beschreibung von Bereinigungsprogrammen (Evictor) im Handbuch *Produktübersicht*. Diese Eigenschaft gilt nur für einen Containerserver.

catalogServiceEndPoints
Gibt die Endpunkte an, die mit dem Katalogservicecluster verbunden werden sollen. Dieser Wert muss im Format `Host:Port<,Host:Port>` angegeben werden, wobei der Hostwert der listenerHost-Wert und der Portwert der listenerPort-Wert des Katalogservers sind. Diese Eigenschaft gilt nur für einen Containerserver.

Eigenschaften des Katalogservice

domainName
Gibt den Domänennamen an, der verwendet wird, um dieses Katalogservice-Grid für Clients eindeutig zu identifizieren, wenn Anforderungen an mehrere Domänen verteilt werden. Diese Eigenschaft gilt nur für den Katalogservice.

enableQuorum
Aktiviert das Quorum für den Katalogservice. Das Quorum wird verwendet, um sicherzustellen, dass die Mehrheit der Member des Katalogservice-Grids verfügbar ist, bevor Änderungen an der Verteilung von Partitionen in den verfügbaren Containerservern zugelassen werden. Zum Aktivieren

des Quorums setzen Sie die Eigenschaft auf `true` oder `enabled`. Der Standardwert ist `disabled`. Diese Eigenschaft gilt nur für den Katalogservice.

catalogClusterEndpoints
Gibt die Endpunkte des Katalogservice-Grids für den Katalogservice an. Diese Eigenschaft gibt die Endpunkte des Katalogservice für das Starten des Katalogservice-Grids an. Verwenden Sie das folgende Format:

```
Servername:Hostname:Clientport:Peer-Port<Servername:Hostname:Clientport:Peer-Port>
```

Diese Eigenschaft gilt nur für den Katalogservice.

heartBeatFrequencyLevel
Gibt an, wie oft Überwachungssignale gesendet werden. Die Überwachungssignalfrequenzstufe ist ein Kompromiss zwischen Ressourcenbelegung und Fehlererkennungszeit. Je häufiger Überwachungssignale gesendet werden, desto mehr Ressourcen werden belegt, aber Fehler werden schneller erkannt. Diese Eigenschaft gilt nur für den Katalogservice.Verwenden Sie einen der folgenden Werte:

- `0`: Gibt eine typische Überwachungssignalfrequenz an. Mit diesem Wert werden Fehler in einer angemessenen Zeit ohne Ressourcenüberbelegung erkannt. (Standardwert)
- `-1`: Gibt eine aggressive Überwachungssignalfrequenz an. Mit diesem Wert werden Fehler zwar schneller erkannt, aber es werden auch mehr Prozessor- und Netzressourcen belegt. Auf dieser Stufe werden fehlende Überwachungssignale schneller erkannt, wenn der Server ausgelastet ist.
- `1`: Gibt eine gelockerte Überwachungssignalfrequenz an. Mit diesem Wert erhöht sich die Zeit für die Erkennung von Fehlern, aber es werden weniger Prozessor- und Netzressourcen belegt.

Sicherheitseigenschaften des Servers

Die Servereigenschaftendatei wird auch verwendet, um die Sicherheit des eXtreme-Scale-Servers zu konfigurieren. Sie verwenden eine einzige Servereigenschaftendatei, um die Basiseigenschaften und die Sicherheitseigenschaften zu definieren.

Allgemeine Sicherheitseigenschaften

securityEnabled
Wenn Sie diese Eigenschaft auf `true` setzen, wird die Sicherheit des Containerservers aktiviert. Der Standardwert ist `false`. Diese Eigenschaft muss der Eigenschaft ″securityEnabled″ entsprechen, die in der an den Katalogserver übergebenen Datei `objectGridSecurity.xml` angegeben ist.

credentialAuthentication
Gibt an, ob dieser Server die Authentifizierung von Berechtigungsnachweisen unterstützt. Wählen Sie einen der folgenden Werte aus:

- `Never`: Der Server unterstützt die Authentifizierung von Berechtigungsnachweisen nicht.
- `Supported`: Der Server unterstützt die Authentifizierung von Berechtigungsnachweisen, wenn auch der Client die Authentifizierung von Berechtigungsnachweisen unterstützt.
- `Required`: Der Client erfordert eine Authentifizierung der Berechtigungsnachweise.

Weitere Informationen zur Authentifizierung von Berechtigungsnachweisen finden Sie im Abschnitt „Anwendungsclientauthentifizierung” auf Seite 312.

Sicherheitseinstellungen auf Transportebene

transportType
Gibt den Transporttyp des Servers an. Verwenden Sie einen der folgenden Werte:

- `TCP/IP`: Gibt an, dass der Server nur TCP/IP-Verbindungen unterstützt.
- `SSL-Supported`: Gibt an, dass der Server TCP/IP- und SSL-Verbindungen (Secure Sockets Layer) unterstützt. (Standardwert)
- `SSL-Required`: Gibt an, dass der Server SSL-Verbindungen erfordert.

SSL-Konfigurationseigenschaften

Alias Gibt den Aliasnamen im Keystore an. Diese Eigenschaft wird verwendet, wenn der Keystore mehrere Schlüsselpaarzertifikate hat und Sie eines der Zertifikate auswählen möchten.

Standardeinstellung: Ohne Wert

contextProvider
Gibt den Namen des Kontextproviders für den Trust-Service an. Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme ein, die anzeigt, dass der Kontextprovidertyp ungültig ist.

Gültige Werte: `IBMJSSE2`, `IBMJSSE`, `IBMJSSEFIPS` usw.

protocol
Gibt den Typ des für den Client zu verwendenden Sicherheitsprotokolls an. Setzen Sie diesen Protokollwert auf der Basis des verwendeten JSSE-Providers (Java Secure Socket Extension). Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme ein, die anzeigt, dass der Protokollwert ungültig ist.

Gültige Werte: `SSL`, `SSLv2`, `SSLv3`, `TLS`, `TLSv1` usw.

keyStoreType
Gibt den Typ des Keystores an.Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme in der Laufzeitumgebung ein.

Gültige Werte: `JKS`, `JCEK`, `PKCS12` usw.

trustStoreType
Gibt den Typ des Truststores an.Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme in der Laufzeitumgebung ein.

Gültige Werte: `JKS`, `JCEK`, `PKCS12` usw.

keyStore
Gibt einen vollständig qualifizierten Pfad zur Keystore-Datei an.

Beispiel:

```
etc/test/security/client.private
```

trustStore
Gibt einen vollständig qualifizierten Pfad zur Truststore-Datei an.

Beispiel:

```
etc/test/security/server.public
```

keyStorePassword
Gibt die Kennwortzeichenfolge für den Keystore an. Sie können diesen Wert verschlüsseln oder den tatsächlichen Wert verwenden.

trustStorePassword
Gibt eine Kennwortzeichenfolge für den Truststore an. Sie können diesen Wert verschlüsseln oder den tatsächlichen Wert verwenden.

clientAuthentication
Wenn Sie diese Eigenschaft auf ″true″ setzen, muss der SSL-Client authentifiziert werden. Die Authentifizierung des SSL-Clients unterscheidet sich von der Authentifizierung des Clientzertifikats. Bei der Authentifizierung des Clientzertifikats wird ein Client auf der Basis der der Zertifizierungskette für eine Benutzer-Registry registriert. Diese Eigenschaft stellt sicher, dass der Server eine Verbindung zum richtigen Server herstellt.

Einstellung ″SecureTokenManager″

Die Einstellung ″SecureTokenManager″ wird verwendet, um die Shared-Secret-Zeichenfolge für die gegenseitige Serverauthentifizierung und das SSO-Token (Single Sign-on) zu schützen. Weitere Informationen finden Sie im Abschnitt „Grid-Sicherheit” auf Seite 310.

secureTokenManagerType
Gibt den Typ der Einstellung ″SecureTokenManager″ an. Sie können eine der folgenden Einstellungen verwenden:

- `none`: Gibt an, dass kein Manager für sichere Token verwendet wird.
- `default`: Gibt an, dass der mit dem Produkt WebSphere eXtreme Scale bereitgestellte Tokenmanager verwendet wird. Sie müssen eine SecureToken-Keystore-Konfiguration angeben.
- `custom`: Gibt an, dass Sie einen eigenen Tokenmanager haben, den Sie mit der Implementierungsklasse ″SecureTokenManager″ angegeben haben.

customTokenManagerClass
Gibt den Namen Ihrer SecureTokenManager-Implementierungsklasse an, wenn Sie `custom` als Wert für die Eigenschaft ″SecureTokenManagerType″ angegeben haben. Die Implementierungsklasse muss einen zu instanzierenden Standardkonstruktor enthalten.

customSecureTokenManagerProps
Gibt die Eigenschaften der angepassten Implementierungsklasse ″SecureTokenManager″ an. Diese Eigenschaft wird nur verwendet, wenn die Eigenschaft ″secureTokenManagerType″ den Wert `custom` hat. Der Wert wird mit der Methode ″setProperties(String)″ im SecureTokenManager-Objekt gesetzt.

Konfiguration eines sicheren Token-Keystores

secureTokenKeyStore
Gibt den Dateipfadnamen für den Keystore an, in dem das Schlüsselpaar aus öffentlichem und privatem Schlüssel und der geheime Schlüssel gespeichert sind.

secureTokenKeyStoreType
Gibt den Keystore-Typ an, z. B. JCKES. Sie können diesen Wert auf der Basis des verwendeten JSSE-Providers (Java Secure Socket Extension) festlegen. Dieser Keystore muss jedoch geheime Schlüssel unterstützen.

secureTokenKeyPairAlias
Gibt den Alias des Schlüsselpaars aus öffentlichem und privatem Schlüssel an, das für die Signatur und Prüfung verwendet wird.

secureTokenKeyPairPassword
Gibt das Kennwort für den Schutz des Schlüsselpaars aus öffentlichem und privatem Schlüssel an, das für Signatur und Prüfung verwendet wird.

secureTokenSecretKeyAlias
Gibt den Alias des geheimen Schlüssels an, der für die Verschlüsselung verwendet wird.

secureTokenSecretKeyPassword
Gibt das Kennwort zum Schutz des geheimen Schlüssels an.

secureTokenCipherAlgorithm
Gibt den verwendeten Algorithmus für die Verschlüsselung an. Sie können diesen Wert auf der Basis des verwendeten JSSE-Providers (Java Secure Socket Extension) festlegen.

secureTokenSignAlgorithm
Gibt den für das Signieren des Objekts verwendeten Algorithmus an. Sie können diesen Wert auf der Basis des verwendeten JSSE-Providers festlegen.

Authentifizierungszeichenfolge

authenticationSecret
Gibt die Shared-Secret-Zeichenfolge für die Abfrage des Servers an. Wenn ein Server gestartet wird, muss er diese Zeichenfolge beim übergeordneten Server oder Katalogserver vorlegen. Wenn die Shared-Secret-Zeichenfolge der Zeichenfolge des übergeordneten Servers entspricht, darf der Server beitreten.

Zugehörige Konzepte

„Transport Layer Security und Secure Sockets Layer" auf Seite 317
WebSphere eXtreme Scale unterstützt TCP/IP und Transport Layer Security/Secure Sockets Layer (TLS/SSL) für die sichere Kommunikation zwischen Client und Server in dynamischen Implementierungsmodellen.

Zugehörige Tasks

„Eigenständige Server von WebSphere eXtreme Scale starten" auf Seite 228
Wenn Sie eine eigenständige Konfiguration von WebSphere eXtreme Scale verwenden, setzt sich die Umgebung aus Katalogservern, Containerservern und eXtreme-Scale-Clientprozessen zusammen. Sie müssen diese Prozesse manuell konfigurieren und starten.

„WebSphere eXtreme Scale mit WebSphere Application Server verwalten" auf Seite 242
Sie können Katalogservice- und Containerserverprozesse in WebSphere Application Server ausführen. Der Prozess zum Konfigurieren dieser Server unterscheidet sich von dem in einer eigenständigen Konfiguration. Der Katalogservice kann automatisch in Servern oder im Deployment Manager von WebSphere Application Server gestartet werden. Der Containerprozess wird gestartet, wenn eine eXtreme-Scale-Anwendung in der Umgebung von WebSphere Application Server implementiert und gestartet wird.

Clienteigenschaftendatei

Sie können eine Eigenschaftendatei erstellen, die auf Ihren Anforderungen für eXtreme-Scale-Clientprozesse basiert.

Musterclienteigenschaftendatei

Sie können die Datei `sampleClient.properties`, die im Verzeichnis *`Stammverzeichnis_von_eXtreme_Scale`*`\properties` enthalten ist, verwenden, um eine eigene Eigenschaftendatei zu erstellen.

Clienteigenschaftendatei angeben

Sie können die Clienteigenschaftendatei mit einer der folgenden Methoden angeben. Wenn Sie eine Einstellung über einen der Einträge weiter hinten in der Liste angeben, wird die vorherige Einstellung überschrieben. Geben Sie beispielsweise einen Systemeigenschaftswert für die Clienteigenschaftendatei an, überschreiben die Eigenschaften in dieser Datei die Werte in der Datei `objectGridClient.properties`, die im Klassenpfad enthalten ist.

1. Sie können irgendeine korrekt benannte Datei angeben, die im Klassenpfad enthalten ist. Das Ablegen dieser Datei im Systemarbeitsverzeichnis wird nicht unterstützt:

   ```
   objectGridClient.properties
   ```

2. Sie können die Datei als Systemeigenschaft in einer eigenständigen Konfiguration oder in einer Konfiguration von WebSphere Application Server angeben. Der Wert kann eine Datei im Systemarbeitsverzeichnis, aber keine Datei im Klassenpfad bezeichnen:

   ```
   -Dobjectgrid.client.props=Dateiname
   ```

3. Sie können die Datei als Korrekturwert mit der Methode "ClientClusterContext.getClientProperties" über das Programm angeben. Das Objekt wird mit Daten aus den Eigenschaftendateien gefüllt. Mit dieser Methode können keine Sicherheitseigenschaften konfiguriert werden.

Clienteigenschaften

preferLocalProcess
: Gibt an, ob der lokale Prozess für das Routing bevorzugt wird. Wenn Sie diese Eigenschaft auf "true" setzen, werden Anforderungen an Shards weitergeleitet, die sich in demselben Prozess wie der Client befinden, sofern dies angebracht ist.

  Standardwert: `true`

preferLocalHost
: Gibt an, ob der lokale Host für das Routing bevorzugt wird. Wenn Sie diese Eigenschaft auf "true" setzen, werden Anforderungen an Shards weitergeleitet, die sich auf demselben Host wie der Client befinden, sofern dies angebracht ist.

  Standardwert: `true`

preferZones
: Gibt eine Liste der bevorzugten Routing-Zonen an. Die angegebenen Zonen werden jeweils durch ein Komma voneinander getrennt: `preferZones=ZoneA,ZoneB,ZoneC`

  Standardwert: Ohne

requestRetryTimeout
: Gibt an, wie lange eine Anforderung wiederholt wird (in Millisekunden). Verwenden Sie einen der folgenden gültigen Werte:

  - Der Wert `0` gibt an, dass die Anforderung schnell fehlschlagen (fail-fast) und die interne Wiederholungslogik übersprungen werden soll.

- Der Wert `-1` zeigt an, dass kein Zeitlimit definiert wurde, d. h., die Anforderungsdauer wird über das Transaktionszeitlimit gesteuert. (Standardwert)
- Ein Wert größer als `0` gibt das Zeitlimit für den Anforderungseingang in Millisekunden an. Ausnahmen, bei denen auch nach einer Wiederholung der Anforderung, keine Wiederherstellung möglich ist, wie z. B. bei der Ausnahme "DuplicateException", werden sofort zurückgegeben. Das Transaktionszeitlimit wird weiterhin als maximale Wartezeit verwendet.

Sicherheitseigenschaften des Clients

Allgemeine Sicherheitseigenschaften

securityEnabled
Aktiviert die Sicherheit für eXtreme-Scale-Clients. Diese Einstellung für die Sicherheitsaktivierung muss mit der Einstellung "securityEnabled" in der Servereigenschaftendatei von WebSphere eXtreme Scale übereinstimmen. Stimmen die Einstellungen nicht überein, tritt eine Ausnahme ein.

Standardwert: `false`

Konfigurationseigenschaften für die Authentifizierung von Berechtigungsnachweisen

credentialAuthentication
Gibt die Unterstützung für die Authentifizierung von Clientberechtigungsnachweisen an. Verwenden Sie einen der folgenden gültigen Werte:

- `Never`: Der Client unterstützt die Authentifizierung von Berechtigungsnachweisen nicht.
- `Supported`: Der Client unterstützt die Authentifizierung von Berechtigungsnachweisen, wenn auch der Server die Authentifizierung von Berechtigungsnachweisen unterstützt. (Standardwert)
- `Required`: Der Client erfordert eine Authentifizierung der Berechtigungsnachweise.

authenticationRetryCount
Gibt an, wie oft die Authentifizierung wiederholt wird, wenn der Berechtigungsnachweis abgelaufen ist. Beim Wert 0 findet keine Wiederholung der Authentifizierungsversuche statt.

Standardwert: `3`

credentialGeneratorClass
Gibt den Namen der Klasse an, die die Schnittstelle "com.ibm.websphere-.objectgrid.security.plugins.CredentialGenerator" implementiert. Diese Klasse wird verwendet, um Berechtigungsnachweise für Clients abzurufen.

Standardwert: Ohne

credentialGeneratorProps
Gibt die Eigenschaften für die Implementierungsklasse "CredentialGenerator" an. Die Eigenschaften werden mit der Methode "setProperties(String)" auf das Objekt gesetzt. Der Wert "credentialGeneratorprops" wird nur verwendet, wenn der Wert der Eigenschaft "credentialGeneratorClass" ungleich null ist.

Konfigurationseigenschaften für die Sicherheit auf Transportebene

transportType
Gibt den Transporttyp des Clients an. Die gültigen Werte sind:

- `TCP/IP`: Gibt an, dass der Client nur TCP/IP-Verbindungen unterstützt.

- `SSL-Supported`: Gibt an, dass der Client TCP/IP- und SSL-Verbindungen (Secure Sockets Layer) unterstützt. (Standardwert)
- `SSL-Required`: Gibt an, dass der Client SSL-Verbindungen erfordert.

SSL-Konfigurationseigenschaften

Alias Gibt den Aliasnamen im Keystore an. Diese Eigenschaft wird verwendet, wenn der Keystore mehrere Schlüsselpaarzertifikate hat und Sie eines der Zertifikate auswählen möchten.

Standardeinstellung: Ohne Wert

contextProvider
Gibt den Namen des Kontextproviders für den Trust-Service an. Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme ein, die anzeigt, dass der Kontextprovidertyp ungültig ist.

Gültige Werte: `IBMJSSE2`, `IBMJSSE`, `IBMJSSEFIPS` usw.

protocol
Gibt den Typ des für den Client zu verwendenden Sicherheitsprotokolls an. Setzen Sie diesen Protokollwert auf der Basis des verwendeten JSSE-Providers (Java Secure Socket Extension). Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme ein, die anzeigt, dass der Protokollwert ungültig ist.

Gültige Werte: `SSL`, `SSLv2`, `SSLv3`, `TLS`, `TLSv1` usw.

keyStoreType
Gibt den Typ des Keystores an. Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme in der Laufzeitumgebung ein.

Gültige Werte: `JKS`, `JCEK`, `PKCS12` usw.

trustStoreType
Gibt den Typ des Truststores an.Wenn Sie einen ungültigen Wert angeben, tritt eine Sicherheitsausnahme in der Laufzeitumgebung ein.

Gültige Werte: `JKS`, `JCEK`, `PKCS12` usw.

keyStore
Gibt einen vollständig qualifizierten Pfad zur Keystore-Datei an.

Beispiel:

```
etc/test/security/client.private
```

trustStore
Gibt einen vollständig qualifizierten Pfad zur Truststore-Datei an.

Beispiel:

```
etc/test/security/server.public
```

keyStorePassword
Gibt die Kennwortzeichenfolge für den Keystore an. Sie können diesen Wert verschlüsseln oder den tatsächlichen Wert verwenden.

trustStorePassword
Gibt eine Kennwortzeichenfolge für den Truststore an. Sie können diesen Wert verschlüsseln oder den tatsächlichen Wert verwenden.

Zugehörige Konzepte

„Transport Layer Security und Secure Sockets Layer" auf Seite 317
WebSphere eXtreme Scale unterstützt TCP/IP und Transport Layer Security/Secure Sockets Layer (TLS/SSL) für die sichere Kommunikation zwischen Client und Server in dynamischen Implementierungsmodellen.

Zugehörige Tasks

„Eigenständige Server von WebSphere eXtreme Scale starten" auf Seite 228
Wenn Sie eine eigenständige Konfiguration von WebSphere eXtreme Scale verwenden, setzt sich die Umgebung aus Katalogservern, Containerservern und eXtreme-Scale-Clientprozessen zusammen. Sie müssen diese Prozesse manuell konfigurieren und starten.

„WebSphere eXtreme Scale mit WebSphere Application Server verwalten" auf Seite 242
Sie können Katalogservice- und Containerserverprozesse in WebSphere Application Server ausführen. Der Prozess zum Konfigurieren dieser Server unterscheidet sich von dem in einer eigenständigen Konfiguration. Der Katalogservice kann automatisch in Servern oder im Deployment Manager von WebSphere Application Server gestartet werden. Der Containerprozess wird gestartet, wenn eine eXtreme-Scale-Anwendung in der Umgebung von WebSphere Application Server implementiert und gestartet wird.

ORB-Eigenschaftendatei

Die Datei `orb.properties` wird für die Übergabe der vom Object Request Broker (ORB) verwendeten Eigenschaften verwendet, um das Transportverhalten des Grids zu ändern.

Position

Sie finden die Datei `orb.properties` im Verzeichnis `java/jre/lib`. Wenn Sie die Datei in einem Verzeichnis `java/jre/lib` von WebSphere Application Server ändern, verwenden auch die Anwendungsserver, die unter dieser Installation konfiguriert sind, die Einstellungen aus dieser Datei.

Basiseinstellungen

Die folgenden Einstellungen bilden eine gute Grundlage, sind aber nicht unbedingt die besten Einstellungen für jede Umgebung. Die Einstellungen helfen Ihnen dabei, eine gute Entscheidung bezüglich der Werte zu treffen, die in Ihrer Umgebung angemessen sind:

```
com.ibm.CORBA.RequestTimeout=30
com.ibm.CORBA.ConnectTimeout=10
com.ibm.CORBA.FragmentTimeout=30
com.ibm.CORBA.ThreadPool.MinimumSize=256
com.ibm.CORBA.ThreadPool.MaximumSize=256
com.ibm.CORBA.ThreadPool.IsGrowable=false
com.ibm.CORBA.ConnectionMultiplicity=1
com.ibm.CORBA.MinOpenConnections=1024
com.ibm.CORBA.MaxOpenConnections=1024
com.ibm.CORBA.ServerSocketQueueDepth=1024
com.ibm.CORBA.FragmentSize=0com.ibm.CORBA.iiop.NoLocalCopies=true
com.ibm.CORBA.NoLocalInterceptors=true
```

Beschreibungen der Eigenschaften

Zeitlimiteinstellungen

Die folgenden Einstellungen beziehen sich auf die Zeit, die der ORB wartet, bevor er Anforderungsoperationen aufgibt.

Anforderungszeitlimit

Eigenschaftsname: com.ibm.CORBA.RequestTimeout

Wert: Ganzzahliger Wert (in Sekunden)

Beschreibung: Gibt an, wie lange (in Sekunden) eine Anforderung auf eine Antwort warten soll, bevor sie aufgibt. Diese Eigenschaft beeinflusst die Zeit, die ein Client für das Failover benötigt, wenn ein Netzausfall eintritt. Wenn Sie einen zu niedrigen Wert für diese Eigenschaft wählen, können nicht beabsichtigte Zeitlimitüberschreitungen bei Anforderungen auftreten. Sie können dies verhindern, indem Sie den Wert für diese Eigenschaft sorgfältig auswählen.

Verbindungszeitlimit

Eigenschaftsname: com.ibm.CORBA.ConnectTimeout

Wert: Ganzzahliger Wert (in Sekunden)

Beschreibung: Gibt an, wie viele Sekunden bei einem Versuch, eine Socket-Verbindung herzustellen, gewartet werden soll, bis der Versuch eingestellt wird. Diese Eigenschaft kann wie das Anforderungszeitlimit die Zeit beeinflussen, die ein Client für das Failover benötigt, wenn ein Netzausfall eintritt. Im Allgemeinen setzen Sie diese Eigenschaft auf einen kleineren Wert als das Anforderungszeitlimit, weil die Zeit zum Herstellen einer Verbindung relativ konstant sein sollte.

Fragmentzeitlimit

Eigenschaftsname: com.ibm.CORBA.FragmentTimeout

Wert: Ganzzahliger Wert (in Sekunden)

Beschreibung: Gibt an, wie lange (in Sekunden) eine Fragmentanforderung auf eine Antwort warten soll, bevor sie aufgibt. Diese Eigenschaft ist der Eigenschaft "Anforderungszeitlimit" ähnlich.

Einstellungen des Thread-Pools

Diese Eigenschaften beschränken die Größe des Thread-Pools auf eine bestimmte Anzahl an Threads. Die Threads werden vom ORB verwendet, um die Serveranforderungen auszugliedern, nachdem sie am Socket empfangen wurden. Wenn Sie zu niedrige Werte für diese Eigenschaften wählen, kann dies zu einer erhöhten Socket-Warteschlangenlänge und zu möglichen Zeitlimitüberschreitungen führen.

Verbindungsmultiplizität

Eigenschaftsname: com.ibm.CORBA.ConnectionMultiplicity

Wert: Ganzzahliger Wert für die Anzahl der Verbindungen zwischen dem Client und dem Server. Der Standardwert ist 1. Die Festlegung eines höheren Werts definiert das Multiplexing für mehrere Verbindungen.**Beschreibung:** Ermöglicht dem ORB, mehrere Verbindungen zu jedem Server herzustellen. In der Theorie sollte die Definition dieses Werts die Parallelität der Verbindungen fördern. In der Praxis profitiert die Leistung nicht von der Festlegung der Verbindungsmultiplizität. Setzen Sie diesen Parameter nicht.

Offene Verbindungen

Eigenschaftsnamen: com.ibm.CORBA.MinOpenConnections, com.ibm-.CORBA.MaxOpenConnections

Wert: Ein ganzzahliger Wert für die Anzahl der Verbindungen.**Beschreibung:** Gibt eine Mindest- und eine maximale Anzahl offener Verbindungen an. Der ORB verwaltet einen Cache mit den Verbindungen, die mit Clients hergestellt wurden. Diese Verbindungen werden gelöscht, wenn der com.ibm.CORBA.MaxOpenConnections-Wert übergeben wird. Das Löschen von Verbindungen kann zu einem mangelhaften Verhalten im Grid führen.

Ist erweiterbar

Eigenschaftsname: com.ibm.CORBA.ThreadPool.IsGrowable

Wert: Die gültigen Werte sind `true` und `false` (Boolean).**Beschreibung:** Wenn Sie die Eigenschaft aktivieren, ist der Thread-Pool, den der ORB für eingehende Anforderungen verwendet, über die vom Pool unterstützte Größe hinaus erweiterbar. Wenn die Poolgröße überschritten wird, werden neue Threads für die Bearbeitung der Anforderungen erstellt, aber die Threads werden nicht in den Pool gestellt.

Länge der Server-Socket-Warteschlange

Eigenschaftsname: com.ibm.CORBA.ServerSocketQueueDepth

Wert: Ein ganzzahliger Wert für die Anzahl der Verbindungen.**Beschreibung:** Gibt die Länge der Warteschlange für eingehende Verbindungen von Clients an. Der ORB reiht eingehende Verbindungen von Clients in Warteschlangen ein. Wenn die Warteschlange voll ist, werden Verbindungen zurückgewiesen. Das Zurückweisen von Verbindungen kann zu einem mangelhaften Verhalten im Grid führen.

Fragmentgröße

Eigenschaftsname: com.ibm.CORBA.FragmentSize

Wert: Eine ganzzahliger Wert, der die Anzahl der Bytes angibt. Der Standardwert ist 1024.**Beschreibung:** Gibt die maximale Paketgröße an, die der ORB verwendet, wenn er eine Anforderung sendet. Wenn eine Anforderung den Grenzwert für die Fragmentgröße überschreitet, wird diese Anforderung in Anforderungsfragmente aufgeteilt, die jeweils separat gesendet und dann im Server erneut assembliert werden. Das Fragmentieren von Anforderungen ist in unzuverlässigen Netzen hilfreich, in denen Paket unter Umständen erneut gesendet werden müssen. Wenn das Netz jedoch zuverlässig ist, kann das Aufteilen von Anforderungen in Fragmente einen erhöhten Aufwand verursachen.

Keine lokalen Kopien

Eigenschaftsname: com.ibm.CORBA.iiop.NoLocalCopies

Wert: Die gültigen Werte sind `true` und `false` (Boolean). **Beschreibung:** Gibt an, ob der ORB nach Referenz übergibt. Der ORB verwendet standardmäßig Aufrufe des Typs ″pass by value″ (Übergeben nach Wert). Aufrufe des Typs ″pass by value″ verursachen zusätzliche Kosten für Garbage-Collection und Serialisierung im Pfad, wenn die Schnittstelle lokal aufgerufen wird. Wenn Sie diese Einstellung auf ″true″ setzen, verwendet der ORB die Methode ″pass by reference″ (Übergeben nach Referenz), die effizienter ist als der Aufruf des Typs ″pass by value″.

Keine lokalen Interceptor

Eigenschaftsname: com.ibm.CORBA.NoLocalInterceptors

Wert: Die gültigen Werte sind `true` und `false` (Boolean). **Beschreibung:** Gibt an, ob der ORB Anforderungs-Interceptor auch dann aufruft, wenn lokale Anforderungen (prozessintern) gestellt werden. Die von WebSphere eXtreme Scale verwendeten Interceptor sind für die Sicherheit und Routenverarbeitung bestimmt und nicht erforderlich, wenn die Anforderung in dem Prozess verarbeitet wird, in dem sie ausgeführt wird. Interceptor zwischen Prozessen sind nur für RPC-Operationen (Remote Procedure Call) erforderlich. Wenn Sie die Eigenschaft "Keine lokalen Interceptor" aktivieren, können Sie den zusätzlichen Aufwand vermeiden, den lokale Interceptor mit sich bringen.

Wenn Sie die Transportsicherheit zwischen ObjectGrid-Clients und -Servern umsetzen möchten, müssen Sie der Datei `orb.properties` weitere Eigenschaften hinzufügen. Weitere Informationen zu diesen Eigenschaften finden Sie in der Beschreibung der Verwendung der Datei `orb.properties` für die Unterstützung der Transportsicherheit im Abschnitt „Transport Layer Security und Secure Sockets Layer" auf Seite 317.

Zugehörige Konzepte

„Transport Layer Security und Secure Sockets Layer" auf Seite 317
WebSphere eXtreme Scale unterstützt TCP/IP und Transport Layer Security/Secure Sockets Layer (TLS/SSL) für die sichere Kommunikation zwischen Client und Server in dynamischen Implementierungsmodellen.

Zugehörige Tasks

„Angepassten Object Request Broker konfigurieren" auf Seite 56
Sie können eine angepasste Version des Object Request Broker (ORB) mit WebSphere eXtreme Scale verwenden, wenn Sie eigenständige Java-SE-Prozesse in Ihrer Umgebung ausführen.

„Object Request Broker mit WebSphere eXtreme Scale-Prozessen verwenden" auf Seite 30
Sie können WebSphere eXtreme Scale mit Anwendungen verwenden, die den Object Request Broker (ORB) direkt in Umgebungen ohne WebSphere Application Server oder WebSphere Application Server Network Deployment verwenden.

Integration mit dem Spring-Framework

Spring ist ein vielfach eingesetztes Framework für die Entwicklung von Java-Anwendungen. WebSphere eXtreme Scale unterstützt den Einsatz von Spring für die Verwaltung von eXtreme-Scale-Transaktionen und die Konfiguration der Clients und Server, aus denen sich das implementierte speicherinterne Daten-Grid zusammensetzt.

Über Spring verwaltete native Transaktionen

Spring unterstützt containerverwaltete Transaktionen, die einem Java-EE-Anwendungsserver gleichen. Der Spring-Mechanismus kann jedoch in verschiedene Implementierungen integriert werden. WebSphere eXtreme Scale unterstützt die Integration eines Transaktionsmanagers, der Spring die Verwaltung der Lebenszyklen von ObjectGrid-Transaktionen ermöglicht. Weitere Einzelheiten finden Sie in den Informationen zu nativen Transaktionen im *Programmierhandbuch*.

Über Spring verwaltete Erweiterungs-Beans und Unterstützung von Namespaces

Spring kann auch in eXtreme Scale integriert werden, um die Definition von Spring-Beans für Erweiterungspunkte und Plug-ins zu ermöglichen. Dieses Feature unterstützt fortgeschrittene Konfigurationen und mehr Flexibilität für die Konfiguration der Erweiterungspunkte.

Zusätzlich zu den über Spring verwalteten Erweiterungs-Beans stellt eXtreme Scale einen Spring-Namespace mit dem Namen ″objectgrid″ bereit. Beans und integrierte Implementierungen sind in diesem Namespace vordefiniert. Dies erleichtert Benutzern die Konfiguration von eXtreme Scale. Weitere Einzelheiten zu diesen Themen und ein Beispiel zum Starten eines eXtreme-Scale-Servers mit Spring-Konfigurationen finden Sie im Abschnitt „Spring-Erweiterungs-Beans und Unterstützung von Namespaces″.

Unterstützung des Geltungsbereichs ″Shard″

Mit der traditionellen Spring-Konfiguration kann eine ObjectGrid-Bean ein Singleton oder ein Prototyp sein. ObjectGrid unterstützt außerdem einen neuen Geltungsbereich, den Geltungsbereich ″Shard″. Wenn eine Bean mit dem Geltungsbereich ″Shard″ definiert wird, kann pro Shard nur eine einzige Bean erstellt werden. Auf alle Anforderungen für Beans mit IDs, die der Bean-Definition im selben Shard entsprechen, wird eine bestimmte Bean-Instanz vom Spring-Container zurückgegeben.

Das folgende Beispiel zeigt eine definierte com.ibm.ws.objectgrid.jpa.plugins-.JPAPropFactoryImpl-Bean mit dem Geltungsbereich ″Shard″. Deshalb wird nur eine einzige Instanz der Klasse ″JPAPropFactoryImpl″ pro Shard erstellt.

```
<bean id="jpaPropFactory" class="com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl" scope="shard" />
```

Spring Web Flow

Spring Web Flow speichert seinen Sitzungsstatus standardmäßig in der HTTP-Sitzung. Wenn in einer Webanwendung die Verwendung von eXtreme Scale für die Sitzungsverwaltung konfiguriert ist, verwendet Spring automatisch eXtreme Scale für das Speichern seines Status und übernimmt die Fehlertoleranz der Sitzung.

Packen

Die Spring-Erweiterungen für eXtreme Scale sind in der Datei `ogspring.jar` enthalten. Diese JAR-Datei (Java-Archiv) muss im Klassenpfad enthalten sein, damit die Spring-Unterstützung funktioniert. Wenn eine Java-EE-Anwendung, die in WebSphere Application Server Network Deployment mit der Erweiterung WebSphere Extended Deployment ausgeführt wird, muss die Anwendung die Datei `spring.jar` und die zugehörigen Dateien in die EAR-Module stellen. Außerdem müssen Sie die Datei `ogspring.jar` an dieselbe Position kopieren.

Spring-Erweiterungs-Beans und Unterstützung von Namespaces

WebSphere eXtreme Scale stellt ein Feature für die Deklaration von POJOs (Plain Old Java Object) als Erweiterungspunkte in der Datei `objectgrid.xml` und eine Methode für die Benennung der Beans und die anschließende Spezifikation des Klassennamens bereit. Normalerweise werden Instanzen der angegebenen Klasse erstellt, und diese Objekte werden dann als Plug-ins verwendet. Jetzt kann

eXtreme Scale das Abrufen von Instanzen dieser Plug-in-Objekte an Spring delegieren. Wenn eine Anwendung Spring verwendet, müssen solche POJOs gewöhnlich mit dem Rest der Anwendung verbunden werden.

In manchen Fällen müssen Sie Spring für die Konfiguration bestimmter Plug-in-Objekte verwenden. Verwenden Sie die folgende Konfiguration als Beispiel:

```
<objectGrid name="Grid">
    <bean id="TransactionCallback" className="com.ibm.websphere.objectgrid.jpa.JPATxCallback">
        <property name="persistenceUnitName" type="java.lang.String"  value="employeePU" />
    </bean>
    ...
</objectGrid>
```

Die integrierte TransactionCallback-Implementierung, die Klasse ″com.ibm.websphere.objectgrid.jpa.JPATxCallback″, ist als TransactionCallback-Klasse konfiguriert. Diese Klasse wird, wie im vorherigen Beispiel gezeigt, mit einer einzigen Eigenschaft, der Eigenschaft ″persistenceUnitName″, konfiguriert. Die Klasse ″JPATxCallback″ besitzt auch das Attribut ″JPAPropertyFactory″, das den Typ ″java.lang.Object″ hat. Die ObjectGrid-XML-Konfiguration unterstützt diesen Typ von Konfiguration nicht.

Die Integration von Spring in eXtreme Scale löst dieses Problem, indem die Bean-Erstellung an das Spring-Framework delegiert wird. Die überarbeitete Konfiguration folgt:

```
<objectGrid name="Grid">
    <bean id="TransactionCallback" className="{spring}jpaTxCallback"/>
    ...
</objectGrid>
```

Die Spring-Datei für das Objekt ″Grid″ enthält die folgenden Informationen:

```
<bean id="jpaTxCallback" class="com.ibm.websphere.objectgrid.jpa.JPATxCallback" scope="shard">
    <property name="persistenceUnitName" value="employeeEMPU"/>
    <property name="JPAPropertyFactory" ref ="jpaPropFactory"/>
</bean>

<bean id="jpaPropFactory" class="com.ibm.ws.objectgrid.jpa.plugins.
JPAPropFactoryImpl" scope="shard">
</bean>
```

Hier ist TransactionCallback mit `{spring}jpaTxCallback` angegeben, und die Beans ″jpaTxCallback″ und ″jpaPropFactory″ werden in der Spring-Datei, wie im vorherigen Beispiel gezeigt, konfiguriert. Mit der Spring-Konfiguration kann eine JPAPropertyFactory-Bean als Parameter des JPATxCallback-Objekts konfiguriert werden.

Standard-Spring-Bean-Factory

Wenn eXtreme Scale ein Plug-in oder eine Erweiterungs-Bean (z. B. ObjectTransformer, Loader, TransactionCallback usw.) mit einem classname-Wert findet, der mit dem Präfix `{spring}` beginnt, verwendet eXtreme Scale den restlichen Teil des Namens als Namen für die Spring-Bean und ruft die Bean-Instanz über die Spring-Bean-Factory ab.

Wenn keine Bean-Factory für ein bestimmtes ObjectGrid registriert wurde, wird standardmäßig versucht, eine Datei `ObjectGridName_spring.xml` zu finden. Wenn Ihr Grid beispielsweise ″Grid″ heißt, hat die XML-Datei den Namen `/Grid_spring.xml`. Diese Datei muss im Klassenpfad oder in einem Verzeichnis `META-INF` im Klassenpfad enthalten sein. Wenn diese Datei gefunden wird, erstellt eXtreme Scale über diese Datei einen Anwendungskontext und über diese Bean-Factory die Beans.

Angepasste Spring-Bean-Factory

WebSphere eXtreme Scale stellt auch eine API ″ObjectGridSpringFactory″ bereit, über die eine Spring-Bean-Factory-Instanz für ein bestimmtes ObjectGrid registriert werden kann. Diese API registriert eine Instanz von BeanFactory mit der folgenden statischen Methode bei eXtreme Scale:

```
void registerSpringBeanFactoryAdapter(String objectGridName, Object
springBeanFactory)
```

Unterstützung von Namespaces

Seit Version 2.0 hat Spring einen Mechanismus für schemabasierte Erweiterungen für das Spring-XML-Basisformat für die Definition und Konfiguration von Beans. ObjectGrid verwendet dieses neue Feature, um ObjectGrid-Beans zu definieren und zu konfigurieren. Mit der Spring-XML-Schemaerweiterung sind einige der integrierten Implementierungen von eXtreme-Scale-Plug-ins und einige ObjectGrid-Beans im Namespace ″objectgrid″ vordefiniert. Wenn Sie die Spring-Konfigurationsdateien schreiben, müssen Sie den vollständigen Klassennamen der integrierten Implementierungen nicht angeben. Stattdessen können Sie die vordefinierten Beans referenzieren.

Außerdem sinkt mit den Attributen der Beans, die im XML-Schema definiert sind, das Risiko, dass falsche Attributnamen angegeben werden. Die XML-Validierung, die auf dem XML-Schema basiert, kann diese Art von Fehlern früher im Entwicklungszyklus abfangen.

Die folgenden Beans sind in den XML-Schemaerweiterungen definiert:

- transactionManager
- register
- server
- catalog
- container
- JPALoader
- JPATxCallback
- JPAEntityLoader
- LRUEvictor
- LFUEvictor
- HashIndex

Diese Beans sind in der XML-Schemadatei ″objectgrid.xsd″ definiert. Diese XSD-Datei wird als Datei `com/ibm/ws/objectgrid/spring/namespace/objectgrid.xsd` in der Datei `ogspring.jar` geliefert. Ausführliche Beschreibungen der XSD-Datei und der in der XSD-Datei definierten Beans finden Sie in den Informationen zur Spring-Deskriptordatei im *Administratorhandbuch*.

Verwenden Sie weiterhin das JPATxCallback-Beispiel aus dem vorherigen Abschnitt. Im vorherigen Abschnitt wurde die JPATxCallback-Bean wie folgt konfiguriert:

```
<bean id="jpaTxCallback" class="com.ibm.websphere.objectgrid.jpa.JPATxCallback" scope="shard">
    <property name="persistenceUnitName" value="employeeEMPU"/>
    <property name="JPAPropertyFactory" ref ="jpaPropFactory"/>
</bean>

<bean id="jpaPropFactory" class="com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl" scope="shard">
</bean>
```

Mit dem Namespace-Feature kann die Spring-XML-Konfiguration wie folgt geschrieben werden:

```
<objectgrid:JPATxCallback id="jpaTxCallback" persistenceUnitName="employeeEMPU"
 jpaPropertyFactory="jpaPropFactory" />

<bean id="jpaPropFactory" class="com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl"
 scope="shard">
</bean>
```

Beachten Sie, dass hier die Klasse ″com.ibm.websphere.objectgrid.jpa-.JPATxCallback″ nicht wie im vorherigen Beispiel angegeben wird, sondern dass die vordefinierte Bean ″objectgrid:JPATxCallback″ direkt verwendet wird. Wie Sie sehen, ist diese Konfiguration weniger ausführlich und in Bezug auf die Fehlerprüfung komfortabler.

Containerserver mit Spring-Erweiterungs-Beans starten

In diesem Beispiel wird veranschaulicht, wie Sie einen ObjectGrid-Server über Spring-verwaltete Erweiterungs-Beans und die Unterstützung von Namespaces von ObjectGrid starten.

ObjectGrid-XML-Datei

Zuerst wird eine sehr einfache ObjectGrid-XML-Datei definiert, die ein einziges ObjectGrid mit dem Namen ″Grid″ und eine einzige Map mit dem Namen ″Test″ enthält. Das ObjectGrid hat ein ObjectGridEventListener-Plug-in mit dem Namen ″partitionListener″ und die Map ″Test″ ein Evictor-Plug-in mit dem Namen ″testLRUEvictor″. Beachten Sie, dass sowohl das ObjectGridEventListener-Plug-in als auch das Evictor-Plug-in mit Spring konfiguriert sind, da ihre Namen ″{spring}″ enthalten:

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
        <objectGrid name="Grid">
     <bean id="ObjectGridEventListener" className="{spring}partitionListener" />
            <backingMap name="Test" pluginCollectionRef="test" />
        </objectGrid>
    </objectGrids>

    <backingMapPluginCollections>
        <backingMapPluginCollection id="test">
            <bean id="Evictor" className="{spring}testLRUEvictor"/>
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

ObjectGrid-XML-Implementierungsdatei

Jetzt wird eine einfache ObjectGrid-XML-Implementierungsdatei erstellt. Sie partitioniert das ObjectGrid in 5 Partitionen, und es ist kein Replikat erforderlich.

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
 xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
    <objectgridDeployment objectgridName="Grid">
        <mapSet name="mapSet" numInitialContainers="1" numberOfPartitions="5" minSyncReplicas="0"
            maxSyncReplicas="1" maxAsyncReplicas="0">
            <map ref="Test"/>
        </mapSet>
    </objectgridDeployment>
</deploymentPolicy>
```

ObjectGrid-Spring-XML-Datei

Jetzt werden die ObjectGrid-Spring-verwalteten Erweiterungs-Beans und die Unterstützung für Namespaces verwendet, um die ObjectGrid-Beans zu konfigurieren.

Die Spring-XML-Datei hat den Namen ″Grid_spring.xml″. Beachten Sie, dass zwei Schemas in der XML-Datei enthalten sind: spring-beans-2.0.xsd ist für die Verwendung der Spring-verwalteten Beans bestimmt und objectgrid.xsd für die im Namespace ″objectgrid″ vordefinierten Beans:

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"
       xsi:schemaLocation="
       http://www.ibm.com/schema/objectgrid
   http://www.ibm.com/schema/objectgrid/objectgrid.xsd
       http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

    <objectgrid:register id="ogregister" gridname="Grid"/>

    <objectgrid:server id="server" isCatalog="true" name="server">
        <objectgrid:catalog host="localhost" port="2809"/>
    </objectgrid:server>

    <objectgrid:container id="container"
   objectgridxml="com/ibm/ws/objectgrid/test/springshard/objectgrid.xml"
       deploymentxml="com/ibm/ws/objectgrid/test/springshard/deployment.xml"
   server="server"/>

    <objectgrid:LRUEvictor id="testLRUEvictor" numberOfLRUQueues="31"/>

    <bean id="partitionListener"
   class="com.ibm.websphere.objectgrid.springshard.ShardListener" scope="shard"/>
</beans>
```

Es wurden 6 in dieser Spring-XML-Datei definiert:

1. *objectgrid:register*: Diese Bean registriert die Standard-Bean-Factory für das ObjectGrid ″Grid″.
2. *objectgrid:server*: Diese Bean definiert einen ObjectGrid-Server mit dem Namen ″server″. Dieser Server stellt auch Katalogservices bereit, da er eine verschachtelte Bean ″objectgrid:catalog″ enthält.
3. *objectgrid:catalog*: Diese Bean definiert einen ObjectGrid-Katalogserviceendpunkt, der auf ″localhost:2809″ gesetzt ist.
4. *objectgrid:container*: Diese Bean definiert einen ObjectGrid-Container mit der angegebenen Objectgrid-XML-Datei und der XML-Implementierungsdatei, die zuvor beschrieben wurden. Die Eigenschaft ″server″ gibt an, in welchem Server dieser Container ausgeführt wird.
5. *objectgrid:LRUEvictor*: Diese Bean definiert einen LRUEvictor mit der einer LRU-Warteschlangenanzahl von 31.
6. *partitionListener*: Diese Bean definiert ein ShardListener-Plug-in. Diese Klasse ist eine Klasse, die von Benutzern integriert wird. Deshalb kann sie die vordefinierten Beans nicht verwenden. Außerdem hat die Bean den Geltungsbereich ″shard″, d. h., es gibt nur eine einzige Instanz dieses ShardListeners pro ObejctGrid-Shard.

Server starten

Das folgende Snippet startet den ObjectGrid-Server, in dem der Containerservice und der Katalogservice ausgeführt werden. Wie zu sehen, ist die einzige Methode, die zum Starten des Servers aufgerufen werden muss, eine Methode ″get″, mit der ein Bean-Container von der Bean-Factory abgerufen wird. Dies vereinfacht die Programmierungskomplexität, weil die meiste Logik in die Spring-Konfiguration verschoben wird:

```
public class ShardServer extends TestCase
{
    Container container;
    org.springframework.beans.factory.BeanFactory bf;

    public void startServer(String cep)
    {
```

```
        try
        {
            bf = new org.springframework.context.support.ClassPathXmlApplicationContext(
                "/com/ibm/ws/objectgrid/test/springshard/Grid_spring.xml", ShardServer.class);
            container = (Container)bf.getBean("container");
        }
        catch (Exception e)
        {
            throw new ObjectGridRuntimeException("Cannot start OG container", e);
        }
    }

    public void stopServer()
    {
        if(container != null)
            container.teardown();
    }
}
```

Spring-XML-Deskriptordatei

Sie können eine Spring-XML-Deskriptordatei verwenden, um eXtreme Scale mit Spring zu konfigurieren und zu integrieren.

In den folgenden Abschnitten finden Sie Definitionen der einzelnen Elemente und Attribute in der Spring-Datei `objectgrid.xsd`. Die Spring-Datei `objectgrid.xsd` befindet sich in der Datei `ogspring.jar` und im ObjectGrid-Namespace `com/ibm/ws/objectgrid/spring/namespace`. Ein Beispiel für das XML-Deskriptorschema finden Sie im Abschnitt „Spring-Datei `objectgrid.xsd`" auf Seite 221.

Element "register"

Verwenden Sie das Element "register", um die Standard-Bean-Factory für das ObjectGrid-Grid zu registrieren.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

id Gibt den Namen des Standard-Bean-Verzeichnisses für ein bestimmtes ObjectGrid an.

gridname
Gibt den Namen der ObjectGrid-Instanz an. Der diesem Attribut zugeordnete Wert muss mit einem gültigen ObjectGrid übereinstimmen, das in der ObjectGrid-Deskriptordatei konfiguriert ist.

```
<register
(1) id="register id"
(2) gridname="ObjectGrid-Name"
/>
```

Element "server"

Verwenden Sie das Element "server", um einen eXtreme-Scale-Server definieren, der einen Container und/oder einen Katalogservice enthalten kann.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

id Gibt den Namen des eXtreme-Scale-Servers an.

tracespec
Gibt den Trace-Typ an und aktiviert die Trace-Erstellung und die Trace-Spezifikation für den Server.

tracefile
Gibt den Pfad und den Namen der zu erstellenden und zu verwendenden Trace-Datei an.

statspec
Gibt die Statistikspezifikation für den Server an.

jmxport
Gibt die Nummer des noch nicht verwendeten Ports an, über den Sie JMX/RMI-Verbindungen aktivieren möchten. JMX unterstützt die Überwachung und Verwaltung über ferne Systeme.

isCatalog
Gibt an, ob der jeweilige Server einen Katalogservice enthält. Der Standardwert ist `false`.

name
Gibt den Namen des Servers an.

```
<server
(1) id="server id"
(2) tracespec="Trace-Spezifikation für den Server"
(3) tracefile="Trace-Datei für den Server"
(4) statspec="Statistikspezifikation für den Server"
(5) jmxport="JMX-Portnummer"
(6) isCatalog="true"|"false"
(7) name="Servername"
/>
```

Element ″catalog″

Verwenden Sie das Element ″catalog″, um Containerservices im Daten-Grid weiterzuleiten.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

host
Gibt den Hostnamen der Workstation an, auf der der Service ausgeführt wird.

port
Gibt die Portnummer an, die in Kombination mit dem Hostnamen den Katalogserviceport bestimmt, zu dem der Client eine Verbindung herstellen kann.

```
<catalog
(1) host="Hostname des Katalogservice"
(2) port="Portnummer des Katalogservice"
/>
```

Element ″container″

Verwenden Sie das Element ″container″, um die Daten selbst zu speichern.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

objectgridxml
Gib den Pfad und den Namen der zu verwendenden XML-Deskriptordatei an, die die Eigenschaften für das ObjectGrid enthält, einschließlich Maps, Sperrstrategie und Plug-ins.

deploymentxml
Gibt den Pfad und den Namen der XML-Datei an, die in Kombination mit der XML-Deskriptordatei Partitionierung, Replikation, Anzahl der anfänglichen Container und weitere Einstellungen bestimmt.

server
Gibt den Server an, auf dem sich der Container befindet.

```
<server
(1) objectgridxml="ObjectGrid-XML-Deskriptordatei"
(2) deploymentxml ="ObjectGrid-XML-Implementierungsdeskriptordatei"
(3) server="Serverreferenz"
/>
```

Element "JPALoader"

Verwenden Sie das Element "JPALoader", um den ObjectGrid-Cache mit einem vorhandenen Back-End-Datenspeicher zu synchronisieren, wenn die API "ObjectMap" verwendet wird.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

entityClassName
Lässt die Verwendung von JPAs wie EntityManager.persist und EntityManager-.find zu. Das Attribut **entityClassName** ist für den JPALoader erforderlich.

preloadPartition
Gibt die Partitionsnummer an, bei der der Preload-Prozess für die Map gestartet wird. Wenn der Wert kleiner als 0 oder größer als (totalNumberOfPartition – 1) ist, wird der Map-Preload-Prozess nicht gestartet.

```
<JPALoader
(1) entityClassName="Name der Entitätsklasse"
(2) preloadPartition ="int"
/>
```

Element "JPATxCallback"

Verwenden Sie das Element "JPATxCallback", um JPA- und ObjectGrid-Transaktionen zu koordinieren.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

persistenceUnitName
Erstellt eine JPA-EntityManagerFactory und sucht die Metadaten der JPA-Entität in der Datei "persistence.xml". Das Attribut **persistenceUnitName** ist erforderlich.

jpaPropertyFactory
Gibt die Factory zum Erstellen einer Map für Persistenzeigenschaften an, mit denen die Standardpersistenzeigenschaften überschrieben werden sollen. Dieses Attribut muss auf eine Bean verweisen.

exceptionMapper
Das ExceptionMapper-Plug-in kann für die Zuordnung von JPA-spezifischen Ausnahmen zu datenbankspezifischen Ausnahmen und umgekehrt verwendet werden. Dieses Attribut muss auf eine Bean verweisen.

```
<JPATxCallback
(1) persistenceUnitName="Name der JPA-Persistenzeinheit"
(2) jpaPropertyFactory ="JPAPropertyFactory-Bean-Referenz"
(3) exceptionMapper="ExceptionMapper-Bean-Referenz"
/>
```

Element ″JPAEntityLoader″

Verwenden Sie das Element ″JPAEntityLoader″, um den ObjectGrid-Cache mit einem vorhandenen Back-End-Datenspeicher zu synchronisieren, wenn die API ″EntityManager″ verwendet wird.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

entityClassName
Lässt die Verwendung von JPAs wie EntityManager.persist und EntityManager-.find zu. Das Attribut **entityClassName** ist für das Element ″JPAEntityLoader″ optional. Wenn das Element nicht konfiguriert ist, wird die entity in der ObjectGrid-Entitäts-Map konfigurierte Entitätsklasse verwendet. Für den ObjectGrid-EntityManager und den JPA-Provider muss dieselbe Klasse verwendet werden.

preloadPartition
Gibt die Partitionsnummer an, bei der der Preload-Prozess für die Map gestartet wird. Wenn der Wert kleiner als 0 oder größer als (totalNumberOfPartition – 1) ist, wird der Map-Preload-Prozess nicht gestartet.

```
<JPAEntityLoader
(1) entityClassName="Name der Entitätsklasse"
(2) preloadPartition ="int"
/>
```

Element ″LRUEvictor″

Verwenden Sie das Element ″LRUEvictor″, um zu entscheiden, welche Einträge entfernt werden, wenn die maximal zulässige Anzahl an Einträgen in einer Map überschritten wird.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

maxSize
Gibt die Gesamtanzahl der Einträge in einer Warteschlange an, bei der das Bereinigungsprogramm (Evictor) eingreifen muss.

sleepTime
Legt das Intervall (in Sekunden) fest, in dem das Bereinigungsprogramm die Map-Warteschlangen überprüft, um festzustellen, ob Aktionen für die Map ausgeführt werden müssen.

numberOfLRUQueues
Gibt an, wie viele Warteschlangen das Bereinigungsprogramm durchsuchen muss, um zu verhindern, dass eine einzige Warteschlange in der Größe der gesamten Map verwendet wird.

useMemoryUsageThresholdEviction
Bestimmt Bereinigungseinstellungen auf der Basis des Speichers, den ein Eintrag belegt. Der Standardwert ist `false`.

```
<LRUEvictor
(1) maxSize="int"
(2) sleepTime ="Sekunden"
(3) numberOfLRUQueues ="int"
(4) useMemoryUsageThresholdEviction ="true"|"false"
/>
```

Element "LFUEvictor"

Verwenden Sie das Element "LFUEvictor", um zu entscheiden, welche Einträge entfernt werden, wenn die maximal zulässige Anzahl an Einträgen in einer Map überschritten wird.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

maxSize
Gibt an, wie viele Einträge insgesamt in jedem Heap-Speicher zulässig sind, bevor das Bereinigungsprogramm eingreifen muss.

sleepTime
Legt das Intervall (in Sekunden) fest, in dem das Bereinigungsprogramm die Heap-Speicher der Maps überprüft, um festzustellen, ob Aktionen für die Map ausgeführt werden müssen.

numberOfHeaps
Gibt an, wie viele Heap-Speicher das Bereinigungsprogramm durchsuchen muss, um zu verhindern, dass ein einziger Heap-Speicher in der Größe der gesamten Map verwendet wird.

useMemoryUsageThresholdEviction
Bestimmt Bereinigungseinstellungen auf der Basis des Speichers, den ein Eintrag belegt.

```
<LFUEvictor
(1) maxSize="int"
(2) sleepTime ="Sekunden"
(3) numberOfHeaps ="int"
(4) useMemoryUsageThresholdEviction ="true"|"false"
/>
```

Element "HashIndex"

Verwenden Sie das Element "HashIndex" mit Java-Reflexion, um in einer Map gespeicherte Objekte zu überwachen, wenn sie aktualisiert werden.

- Anzahl der Vorkommen: 0 bis viele
- Untergeordnetes Element: Ohne

Attribute

name
Gibt den Namen des Index an, der für jede Map eindeutig sein muss.

attributeName
Gibt den Namen des zu indexierenden Attributs an. Bei Feldzugriffsindizes entsprechen die Attributnamen den Feldnamen. Bei Eigenschaftszugriffsindizes sind die Attributnamen die JavaBean-kompatiblen Eigenschaftsnamen.

rangeIndex
Gibt an, ob die Bereichsindexierung aktiviert ist. Der Standardwert ist `false`.

fieldAccessAttribute
Wird für Maps verwendet, die keine Entitäts-Maps sind. Die Getter-Methode

wird für den Zugriff auf die Daten verwendet. Der Standardwert ist `false`. Wenn diese Einstellung den Wert `true` hat, wird direkt über die Felder auf das Objekt zugegriffen.

POJOKeyIndex
Wird für Maps verwendet, die keine Entitäts-Maps sind. Der Standardwert ist `false`. Wenn Sie den Wert `true` angeben, überwacht der Index das Objekt im Schlüsselteil der Map, was hilfreich ist, wenn der Schlüssel ein zusammengesetzter Schlüssel ist und im Wert kein Schlüssel integriert ist. Wenn diese Einstellung nicht angegeben oder auf `false` gesetzt wird, überwacht der Index selbst das Objekt im Wertteil der Map.

```
<HashIndex
(1) name="Indexname"
(2) attributeName="Attributname"
(3) rangeIndex ="true"|"false"
(4) fieldAccessAttribute ="true"|"false"

(5) POJOKeyIndex ="true"|"false"
/>
```

Spring-Datei `objectgrid.xsd`

Verwenden Sie die Spring-Datei `objectgrid.xsd` für die Integration von eXtreme Scale in Spring, um eXtreme-Scale-Transaktionen zu verwalten und Clients und Server zu konfigurieren.

Beschreibungen der Elemente und Attribute, die in der Spring-Datei `objectgrid.xsd` definiert werden, finden Sie im Abschnitt „Spring-XML-Deskriptordatei” auf Seite 216.

Spring-Datei `objectgrid.xsd`

```
<xsd:schema xmlns="http://www.ibm.com/schema/objectgrid"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:beans="http://www.springframework.org/schema/beans"
 targetNamespace="http://www.ibm.com/schema/objectgrid"
 elementFormDefault="qualified"
 attributeFormDefault="unqualified">

 <xsd:import namespace="http://www.springframework.org/schema/beans" />

  <xsd:element name="transactionManager">
  <xsd:complexType>
   <xsd:attribute name="id" type="xsd:ID" />
  </xsd:complexType>
 </xsd:element>

  <xsd:element name="register">
  <xsd:complexType>
   <xsd:attribute name="id" type="xsd:ID" />
   <xsd:attribute name="gridname" type="xsd:string" />
  </xsd:complexType>
 </xsd:element>

  <xsd:element name="server">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element ref="catalog" />
   </xsd:choice>
   <xsd:attribute name="id" type="xsd:ID" />
   <xsd:attribute name="tracespec" type="xsd:string" />
   <xsd:attribute name="tracefile" type="xsd:string" />
   <xsd:attribute name="statspec" type="xsd:string" />
   <xsd:attribute name="jmxport" type="xsd:integer" />
   <xsd:attribute name="isCatalog" type="xsd:boolean" />
   <xsd:attribute name="name" type="xsd:string" />
  </xsd:complexType>
  </xsd:element>

  <xsd:element name="catalog">
   <xsd:complexType>
    <xsd:attribute name="host" type="xsd:string" />
    <xsd:attribute name="port" type="xsd:integer" />
   </xsd:complexType>
  </xsd:element>
```

```
 <xsd:element name="container">
 <xsd:complexType>
   <xsd:attribute name="id" type="xsd:ID" />
   <xsd:attribute name="objectgridxml" type="xsd:string" />
   <xsd:attribute name="deploymentxml" type="xsd:string" />
   <xsd:attribute name="server" type="xsd:string" />
  </xsd:complexType>
</xsd:element>

  <xsd:element name="JPALoader">
  <xsd:complexType>
   <xsd:attribute name="id" type="xsd:ID" />
   <xsd:attribute name="entityClassName" type="xsd:string" />
   <xsd:attribute name="preloadPartition" type="xsd:integer" />
  </xsd:complexType>
</xsd:element>

<xsd:element name="JPATxCallback">
 <xsd:complexType>
  <xsd:attribute name="id" type="xsd:ID" />
  <xsd:attribute name="persistenceUnitName" type="xsd:string" />
   <xsd:attribute name="jpaPropertyFactory" type="xsd:string" />
   <xsd:attribute name="exceptionMapper" type="xsd:string" />
  </xsd:complexType>
</xsd:element>

<xsd:element name="JPAEntityLoader">
 <xsd:complexType>
  <xsd:attribute name="id" type="xsd:ID />
  <xsd:attribute name="entityClassName" type="xsd:string" />
  <xsd:attribute name="preloadPartition" type="xsd:integer" />
 </xsd:complexType"
</xsd:element>

<xsd:element name="LRUEvictor">
 <xsd:complexType>
  <xsd:attribute name="id" type="xsd:ID" />
  <xsd:attribute name="maxSize" type="xsd:integer" />
  <xsd:attribute name="sleepTime" type="xsd:integer" />
  <xsd:attribute name="numberOfLRUQueues" type="xsd:integer" />
  <xsd:attribute name="useMemoryUsageThresholdEviction" type="xsd:boolean" />
 </xsd:complexType>
</xsd:element>

<xsd:element name="LFUEvictor">
 <xsd:complexType>
  <xsd:attribute name="id" type="xsd:ID" />
  <xsd:attribute name="maxSize" type="xsd:integer" />
  <xsd:attribute name="sleepTime" type="xsd:integer" />
  <xsd:attribute name="numberOfHeaps" type="xsd:integer" />
  <xsd:attribute name="useMemoryUsageThresholdEviction" type="xsd:boolean" />
 </xsd:complexType>
</xsd:element>

<xsd:element name="HashIndex">
 <xsd:complexType>
  <xsd:attribute name="id" type="xsd:ID" />
  <xsd:attribute name="name" type="xsd:string" />
  <xsd:attribute name="attributeName" type="xsd:string" />
  <xsd:attribute name="rangeIndex" type="xsd:boolean" />
  <xsd:attribute name="fieldAccessAttribute" type="xsd:boolean" />
  <xsd:attribute name="POJOKeyIndex" type="xsd:boolean" />
 </xsd:complexType"
</xsd:element>

</xsd:schema>
```

WebSphere Real Time verwenden

Sie können WebSphere Real Time mit WebSphere eXtreme Scale verwenden. Durch die Aktivierung von WebSphere Real Time erreichen Sie vorhersehbarere Garbage-Collection mit stabilen, konsistenten Antwortzeiten und Transaktionsdurchsätzen in einer eigenständigen eXteme-Scale-Umgebung.

Gründe für die Verwendung von WebSphere Real Time

WebSphere eXtreme Scale erstellt viele temporäre Objekte für jede Transaktion. Die temporären Objekte beziehen sich auf Anforderungen, Antworten, Protokollfolgen und Sitzungen. Wenn Sie WebSphere Real Time nicht verwenden, kann die Antwortzeit auf mehrere Hundert Millisekunden ansteigen. Der Einsatz von WebSphere Real Time mit WebSphere eXtreme Scale kann die Effizienz der Garba-

ge-Collection steigern und die Antwortzeit auf 10 % der Antwortzeit in der eigenständigen Konfiguration verringern.

WebSphere Real Time aktivieren

Installieren SIe WebSphere Real Time und die eigenständige Konfiguration von WebSphere eXtreme Scale auf den Computern, auf denen Sie eXtreme Scale ausführen möchten. Setzen Sie die Umgebungsvariable JAVA_HOME so, dass sie auf eine Standard-JRE (Java SE Runtime Environment) zeigt.

Setzen Sie die Umgebungsvariable JAVA_HOME so, dass sie auf das installierte Produkt WebSphere Real Time zeigt. Aktivieren Sie WebSphere Real Time anschließend wie folgt.

1. Editieren Sie die Datei `objectgridRoot/bin/setupCmdLine.sh | .bat` der eigenständigen Installation, indem Sie das Kommentarzeichen aus der folgenden Zeile entfernen:

   ```
   WXS_REAL_TIME_JAVA="-Xrealtime -Xgcpolicy:metronome
   -Xgc:targetUtilization=80"
   ```

2. Speichern Sie die Datei.

Jetzt haben Sie WebSphere Real Time aktiviert. Wenn Sie WebSphere Real Time inaktivieren möchten, können Sie der Zeile das Kommentarzeichen wieder hinzufügen.

Bewährte Verfahren

Wenn Sie WebSphere WebSphere Real Time einsetzen, sind die Antwortzeiten von eXtreme-Scale-Transaktionen vorhersehbarer. Die Ergebnisse zeigen, dass sich die Abweichung der Antwortzeit einer eXtreme-Scale-Transaktion mit WebSphere Real Time im Vergleich zum Standard-Java mit dem Standard-Garbage-Collector erheblich verbessert. Die Aktivierung von WebSphere Real Time mit eXtreme Scale ist optimal, wenn Stabilität und Antwortzeiten Ihrer Anwendung von entscheidender Bedeutung sind.

Die in diesem Abschnitt beschriebenen bewährten Verfahren verdeutlichen, wie WebSphere eXtreme Scale durch Optimierung und Codeverfahren für die erwartete Last effizienter gemacht werden kann.

- Legen Sie die richtige Prozessorbelegungsstufe für Ihre Anwendung und den Garbage-Collector fest.

  WebSphere Real Time bietet die Möglichkeit, die Prozessorbelegung zu steuern, so dass die Auswirkungen der Garbage-Collection auf Ihre Anwendung kontrolliert und minimiert werden. Verwenden Sie den Parameter `-Xgc:targetUtilization=NN`, um `NN` Prozent der Prozessorkapazität festzulegen, die alle 20 Sekunden von Ihrer Anwendung belegt werden. Der Standardwert für WebSphere eXtreme Scale ist 80 %, aber Sie können das Script in der Datei `objectgridRoot/bin/setupCmdLine.sh` ändern und eine andere Zahl festlegen, wie z. B. `70`, womit Sie dem Garbage-Collector mehr Prozessorkapazität zuweisen. Implementieren Sie genügend Server, um die Prozessorbelegung für Ihre Anwendungen unter 80 % zu halten.
- Legen Sie einen höheren Wert für die Heap-Speichergröße fest.

  WebSphere Real Time belegt mehr Hauptspeicher als reguläres Java. Planen Sie WebSphere eXtreme Scale deshalb mit einem größeren Heap-Speicher, und legen Sie die Heap-Speichergröße beim Start der Katalogserver und Container mit dem Parameter `–jvmArgs –XmxNNNM` im Befehl ogStartServer fest. Sie können den Para-

meter –jvmArgs –Xmx500M beispielsweise verwenden, um Katalogserver zu starten und eine entsprechende Hauptspeichergröße zum Starten der Container zu verwenden. Sie können die Hauptspeichergröße auf 60-70 % der erwarteten Datenmenge pro JVM setzen. Wenn Sie diesen Wert nicht festlegen, kann ein Fehler des Typs ″OutOfMemoryError″ auftreten. Optional können Sie auch den Parameter –jvmArgs –Xgc:noSynchronousGCOnOOM verwenden, um ein nicht deterministisches Verhalten zu verhindern, wenn in der JVM eine abnormale Speicherbedingung auftritt.

- Threads für die Garbage-Collection anpassen.

  WebSphere eXtreme Scale erstellt eine Vielzahl temporärer Objekte für jede Transaktion und jeden RPC-Thread (Remote Procedure Call). Die Garbage-Collection bietet eine bessere bessere Leistung, wenn Ihr Computer genügtend Prozessorzyklen besitzt. Die Standardanzahl der Threads ist 1. Sie können die Anzahl der Threads mit dem Argument –Xgcthreads n anpassen. Der vorgeschlagene Wert für dieses Argument ist die Anzahl der verfügbaren Kerne unter Berücksichtigung der Anzahl der Java Virtual Machines pro Computer.

- Leistung für Anwendungen mit kurzer Laufzeit mit WebSphere eXtreme Scale anpassen.

  WebSphere Real Time ist für Anwendungen mit langer Laufzeit optimiert. Gewöhnlich müssen Transaktionen von WebSphere eXtreme Scale kontinuierlich über einen Zeitraum von two Stunden hinweg ausgeführt werden, um zuverlässige Leistungsdaten zu erhalten. Sie können den Parameter –Xquickstart verwenden, um die Leistung Ihrer Anwendungen mit kurzer Laufzeit zu verbessern. Dieser Parameter weist den JIT-Compiler an, die untere Stufe der Optimierung zu verwenden.

- Clientwarteschlange von WebSphere eXtreme Scale und Clientvermittlung von WebSphere eXtreme Scale minimieren

  Der Hauptvorteil der Verwendung von WebSphere eXtreme Scale mit WebSphere Real Time ist eine hoch zuverlässige Transaktionsantwortzeit, die gewöhnlich erhebliche Verbesserungen bei der Abweichung der Transaktionsantwortzeiten zur Folge hat. Alle in die Warteschlange eingereihten Clientanforderungen und Clientanforderungsvermittlungen über andere Software wirken sich auf die Antwortzeit aus, die außerhalb der Kontrolle von WebSphere Real Time und WebSphere eXtreme Scale liegt. Sie sollten müssen Ihre Thread- und Socket-Parameter ändern, um eine stabile und gleichmäßige Last ohne größere Verzögerungen zu erzielen und die Länge der Warteschlangen zu verringern.

- Anwendungen von WebSphere eXtreme Scale so schreiben, dass sie das Threading von WebSphere Real Time verwenden

  Sie können ohne Änderung Ihrer Anwendung hoch zuverlässige Transaktionsantwortzeiten in WebSphere eXtreme Scale mit erheblichen Verbesserungen bei der Antwortzeitabweichung erreichen. Außerdem können Sie Threading für Ihre transaktionsorientierten Anwendungen (von regulären Java-Threads zu Realtime-Threads) nutzen, das Ihnen eine bessere Steuerung der Thread-Prioritäten und der Planung bietet.

  Ihre Anwendung enthält derzeit den folgenden Code:

  ```
  public class WXSCacheAppImpl extends Thread implements WXSCacheAppIF
  ```

  Optional können Sie diesen Code durch Folgenden ersetzen.

  ```
  public class WXSCacheAppImpl extends RealtimeThread implements
  WXSCacheAppIF
  ```

Kapitel 7. Umgebung verwalten

Die Verwaltung der Umgebung umfasst sowohl im eigenständigen Modus als auch in WebSphere Application Server das Starten und Stoppen von Servern. Sie können WebSphere eXtreme Scale auch als Sitzungsmanager in einer Umgebung mit WebSphere Application Server verwenden.

Warum und wann dieser Vorgang ausgeführt wird

Servertypen

In WebSphere eXtreme Scale gibt es zwei Typen von Servern: *Katalogserver* und *Containerserver*. Katalogserver steuern die Verteilung von Shards und erkennen und überwachen die Containerserver. Mehrere Katalogserver zusammen bilden den *Katalogservice*. Containerserver sind die Java Virtual Machines, in denen die Anwendungsdaten für das Grid gespeichert werden.

Eigenständiger Modus

Eigenständiger Modus bezeichnet eine WebSphere eXtreme Scale-Konfiguration, die eigenständig, d. h. ohne andere Anwendungsserverprodukte, ausgeführt wird.

Ausführung in WebSphere Application Server

Wenn Sie WebSphere eXtreme Scale in WebSphere Application Server ausführen, werden Katalogserver automatisch in Servern von WebSphere Application Server gestartet. Zum Starten von Containerservern müssen Sie Ihre Anwendung mit ObjectGrid-XML-Dateien packen und implementieren.

Verfügbarkeit eines ObjectGrids festlegen

Der Verfügbarkeitsstatus einer ObjectGrids-Instanz bestimmt, welche Anforderungen zu einer bestimmten Zeit verarbeitet werden können.

Es gibt vier Verfügbarkeitsstatus:

- ONLINE
- QUIESCE
- OFFLINE
- PRELOAD

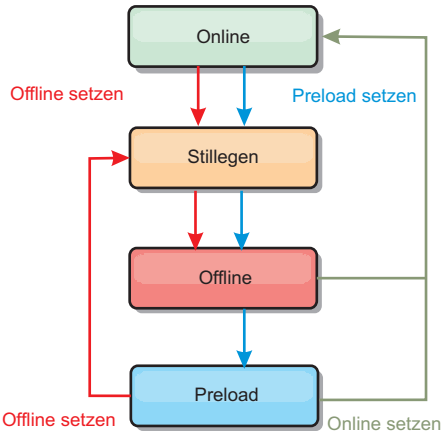


Abbildung 5. Verfügbarkeitsstatus eines ObjectGrids

Verfügbarkeitsstatus festlegen

Standardmäßig hat ein ObjectGrid den Verfügbarkeitsstatus ONLINE. Ein ObjectGrid mit dem Verfügbarkeitsstatus ONLINE kann alle Anforderungen eines typischen eXtreme-Scale-Clients verarbeiten. Anforderungen von Preload-Clients werden jedoch zurückgewiesen, wenn das ObjectGrid den Status ONLINE hat.

Der Status QUIESCE (Stilllegen) ist ein Übergangsstatus. Ein ObjectGrid mit dem Verfügbarkeitsstatus QUIESCE wird bald in den Status OFFLINE versetzt. Wenn ein ObjectGrid den Status QUIESCE hat, können ausstehende Transaktionen verarbeitet werden. Neue Transaktionen werden jedoch zurückgewiesen. Ein ObjectGrid kann bis zu 30 Sekunden im Status QUIESCE verbleiben. Danach wird der Verfügbarkeitsstatus in OFFLINE geändert.

Ein ObjectGrid im Status OFFLINE weist alle Transaktionen zurück.

Der Status PRELOAD kann verwendet werden, um Daten von einem Preload-Client in ein ObjectGrid zu laden. Während das ObjectGrid den Status PRELOAD hat, kann nur ein Preload-Client Transaktionen im ObjectGrid festschreiben. Alle anderen Transaktionen werden zurückgewiesen.

Verwenden Sie die Schnittstelle "StateManager", um den Verfügbarkeitsstatus eines ObjectGrids festzulegen. Mit der Schnittstelle "StateManager" kann der Verfügbarkeitsstatus von Client-ObjectGrids und serverseitigen ObjectGrids geändert werden. Der folgende Code veranschaulicht, wie der Verfügbarkeitsstatus eines Client-ObjectGrids geändert wird:

```
ClientClusterContext client = ogManager.connect("localhost:2809", null, null);
ObjectGrid myObjectGrid = ogManager.getObjectGrid(client, "myObjectGrid");
StateManager stateManager = StateManagerFactory.getStateManager();
stateManager.setObjectGridState(AvailabilityState.OFFLINE, myObjectGrid);
```

Jedes Shard des ObjectGrids nimmt den gewünschten Status an, wenn die Methode "setObjectGridState" in der Schnittstelle "StateManager" aufgerufen wird. Wenn die Methode zurückkehrt, sollten alle Shards im ObjectGrid den richtigen Status haben.

Verwenden Sie ein ObjectGridEventListener-Plug-in, um den Verfügbarkeitsstatus eines serverseitigen ObjectGrids zu ändern. Ändern Sie den Verfügbarkeitsstatus eines serverseitigen ObjectGrids nur, wenn das ObjectGrid eine einzige Partition

hat. Falls das ObjectGrid mehrere Partitionen hat, wird die Methode ″shardActivated″ für jedes primäre Shard aufgerufen, was zu überflüssigen Aufrufen zum Ändern des ObjectGrid-Status führt.

```
public class OGListener implements ObjectGridEventListener,
   ObjectGridEventGroup.ShardEvents {
    public void shardActivated(ObjectGrid grid) {
            StateManager stateManager = StateManagerFactory.getStateManager();
            stateManager.setObjectGridState(AvailabilityState.PRELOAD, grid);
    }
}
```

Da der Status QUIESCE ein Übergangsstatus ist, können Sie die Schnittstelle ″StateManager″ nicht verwenden, um ein ObjectGrid in den Status QUIESCE zu versetzen. Ein ObjectGrid nimmt diesen Status auf seinem Weg zum Status OFFLINE vorübergehend an.

Verfügbarkeitsstatus abrufen

Verwenden Sie die Methode ″getObjectGridState″ der Schnittstelle ″StateManager″, um den Verfügbarkeitsstatus eines bestimmten ObjectGrids abzurufen.

```
StateManager stateManager = StateManagerFactory.getStateManager();
AvailabilityState state = stateManager.getObjectGridState(inventoryGrid);
```

Die Methode ″getObjectGridState″ wählt ein zufälliges primäres Shard im ObjectGrid aus und gibt dessen Verfügbarkeitsstatus zurück. Da alle Shards eines ObjectGrids denselben Verfügbarkeitsstatus haben bzw. auf dem Übergang zu demselben Verfügbarkeitsstatus sein sollten, gibt diese Methode ein akzeptables Ergebnis für den aktuellen Verfügbarkeitsstatus des ObjectGrids zurück.

Erforderliche Verfügbarkeitsstatus für verschiedene Anforderungen

Eine Anforderung wird zurückgewiesen, wenn ein ObjectGrid nicht den erforderlichen Verfügbarkeitsstatus hat, der diese Anforderung unterstützt. Es wird eine Ausnahme des Typs ″AvailabilityException″ ausgegeben, wenn eine Anforderung zurückgewiesen wird.

Attribut ″initialState″

Sie können das Attribut ″initialState″ in einem ObjectGrid verwenden, um dessen Anfangsstatus anzugeben. Nach Abschluss der Initialisierung ist ein ObjectGrid normalerweise für Routing bereit. Der Status kann später geändert werden, um zu verhindern, dass Datenverkehr an ein ObjectGrid weitergeleitet wird. Wenn das ObjectGrid initialisiert werden muss, aber nicht sofort verfügbar ist, können Sie das Attribut ″initialState″ verwenden.

Das Attribut ″initialState″ wird in der XML-Konfigurationsdatei des ObjectGrids definiert. Der Standardstatus ist ONLINE. Die gültigen Werte sind:

- ONLINE (Standardeinstellung)
- PRELOAD
- OFFLINE

Weitere Informationen finden Sie in der API-Dokumentation zu AvailabilityState.

Wenn das Attribut ″initialState″ in einem ObjectGrid definiert wird, muss der Status explizit auf ONLINE zurückgesetzt werden, oder das ObjectGrid bleibt nicht verfügbar. Es werden Ausnahmen des Typs ″AvailabilityException″ ausgelöst.

Attribut ″initialState″ für das vorherige Laden verwenden

Wenn das ObjectGrid vorher mit Daten geladen wird (Preload), kann es einen Zeitraum zwischen Verfügbarkeit des ObjectGrids und Wechseln in einen Preload-Status geben, in dem Clientdatenverkehr blockiert werden kann. Um diesen Zeitraum zu verhindern, kann der Anfangsstatus eines ObjectGrids auf PRELOAD gesetzt werden. Das ObjectGrid setzt zwar die erforderlichen Initialisierungsprozesse fort, blockiert den Datenverkehr aber so lange, bis sich der Status ändert und der Preload-Prozess durchgeführt werden kann.

Die Status PRELOAD und OFFLINE blockieren zwar beide den Datenverkehr, aber Sie müssen den Status PRELOAD verwenden, wenn Sie einen Preload-Prozess einleiten möchten.

Verhalten beim Failover und Lastausgleich

Wenn ein Replikat in ein primäres Shard hochgestuft wird, verwendet es nicht die initialState-Einstellung. Wenn das primäre Shard zur Neuverteilung verschoben wird, wird die initialState-Einstellung nicht verwendet, weil die Daten an die neue primäre Position kopiert werden, bevor der Verschiebevorgang durchgeführt wird. Wenn keine Replikation konfiguriert ist, übernimmt das primäre Shard den initialState-Wert, wenn ein Failover stattfindet und ein neues primäres Shards verteilt werden muss.

Eigenständige Server von WebSphere eXtreme Scale starten

Wenn Sie eine eigenständige Konfiguration von WebSphere eXtreme Scale verwenden, setzt sich die Umgebung aus Katalogservern, Containerservern und eXtreme-Scale-Clientprozessen zusammen. Sie müssen diese Prozesse manuell konfigurieren und starten.

Vorbereitungen

Sie können Server von WebSphere eXtreme Scale in einer Umgebung ohne WebSphere Application Server starten. Wenn Sie WebSphere Application Server verwenden, lesen Sie den Abschnitt „WebSphere eXtreme Scale mit WebSphere Application Server verwalten” auf Seite 242.

Nächste Maßnahme

Stoppen Sie Ihre eXtreme-Scale-Prozesse. Weitere Informationen finden Sie im Abschnitt „Eigenständige eXtreme-Scale-Server stoppen” auf Seite 239.

Zugehörige Konzepte

„Sichere eXtreme-Scale-Server starten und stoppen" auf Seite 327
Servers müssen für die Implementierungsumgebung häufig sicher sein, und dies erfordert eine spezielle Konfiguration.

Zugehörige Verweise

„Servereigenschaftendatei" auf Seite 197
Die Servereigenschaftendatei enthält verschiedene Eigenschaften, mit denen die verschiedenen Einstellungen für Ihren Server definiert werden, z. B. Trace-Einstellungen, Protokollierung und Sicherheitskonfiguration. Die Servereigenschaftendatei wird vom Katalogservice und von den Containerservern verwendet.

„Clienteigenschaftendatei" auf Seite 203
Sie können eine Eigenschaftendatei erstellen, die auf Ihren Anforderungen für eXtreme-Scale-Clientprozesse basiert.

„Referenz der Eigenschaftendatei" auf Seite 196
Servereigenschaftendateien enthalten Einstellungen für die Ausführung der Katalogserver und Containerserver. Sie können eine Servereigenschaftendatei für eine eigenständige Konfiguration oder eine Konfiguration von WebSphere Application Server angeben. Clienteigenschaftendateien enthalten Einstellungen für Ihren Client.

„Script ″startOgServer″" auf Seite 235
Das Script ″startOgServer″ startet Server. Sie können beim Starten Ihrer Server eine Vielzahl von Parametern verwenden, um die Trace-Erstellung zu aktivieren, Portnummern anzugeben usw.

„Protokolle und Trace" auf Seite 303
Sie können Protokolle und Trace verwenden, um Ihre Umgebung zu überwachen und Fehler zu beheben. Protokolle werden je nach Konfiguration an unterschiedlichen Positionen gespeichert. Möglicherweise müssen Sie einen Trace für einen Server bereitstellen, wenn Sie mit der IBM Unterstützungsfunktion zusammenarbeiten.

Katalogservice in einer eigenständigen Umgebung starten

Sie müssen den Katalogservice manuell starten, wenn Sie eine verteilte Umgebung von WebSphere eXtreme Scale ohne WebSphere Application Server verwenden.

Vorbereitungen

Wenn Sie WebSphere Application Server verwenden, wird der Katalogservice automatisch in einem der vorhandenen Prozesse gestartet. Weitere Informationen finden Sie im Abschnitt Katalogservice in WebSphere Application Server starten.

Warum und wann dieser Vorgang ausgeführt wird

Der Katalogservice kann in einem einzelnen Prozess ausgeführt werden, oder es können mehrere Katalogserver zum Katalogserver-Grid zusammengefasst werden. Ein Katalogserver-Grid ist in einer Produktionsumgebung für hohe Verfügbarkeit erforderlich. Weitere Informationen zum Konfigurieren eines Katalogserver-Grids finden Sie in den Informationen zum Clustering eines Katalogservice im Handbuch *Produktübersicht*. Der Katalogservice wird, unabhängig davon, ob er in einem Grid oder in einem einzelnen Prozess ausgeführt wird, über das Script `startOgServer` gestartet. Außerdem können Sie zusätzliche Scriptparameter angeben, um den ORB (Object Request Broker) an eine bestimmte Host/Port-Kombination zu binden, die Domäne anzugeben oder die Sicherheit zu aktivieren.

Wenn Sie den Startbefehl aufrufen, verwenden Sie auf UNIX-Plattformen das Script `startOgServer.sh` und unter Windows das Script `startOgServer.bat`.

- **Einzelnen Katalogserverprozess starten**

  Geben Sie zum Starten eines einzelnen Katalogservers die folgenden Befehle in der Befehlszeile ein:

  1. Navigieren Sie wie folgt zum Verzeichnis "bin":

     ```
     cd ObjectGrid-Stammverzeichnis/bin
     ```

  2. Führen Sie den Befehl "startOgServer" aus:

     ```
     startOgServer.bat|sh catalogServer
     ```

  Eine Liste aller verfügbaren Befehlszeilenparameter finden Sie im Abschnitt „Script "startOgServer"" auf Seite 235. Wenn Sie den Katalogservice in einer Produktionsumgebung ausführen, verwenden Sie keine einzelne Java Virtual Machine (JVM). Wenn der Katalogservice fehlschlägt, können keine neuen Clients Anforderungen an die implementierte eXtreme-Scale-Umgebung weiterleiteten, und es können keine neuen ObjectGrid-Instanzen zur Domäne hinzugefügt werden. Aus diesen Gründen sollten Sie eine Gruppe von Java Virtual Machines starten, um ein Katalogserver-Grid auszuführen.

- **Katalogserver-Grid mit mehreren Prozessen starten**

  Wenn Sie eine Gruppe von Servern zu Ausführen eines Katalogservice starten möchten, müssen Sie die Option **-catalogServiceEndPoints** im Script `startOgServer` verwenden. Dieses Argument akzeptiert eine Liste mit Katalogserviceendpunkten im Format `Servername:Hostname:Clientport:Peer-Port`. Jedes Attribut ist wie folgt definiert:

  serverName
  Gibt einen Namen an, der den Prozess identifiziert, den Sie starten.

  hostName
  Gibt den Hostnamen des Computers an, auf dem der Server gestartet wird.

  clientPort
  Gibt den Port an, der für die Peer-Kommunikation im Katalog-Grid verwendet wird.

  peerPort
  Gibt den Port an, der für die Peer-Kommunikation im Katalog-Grid verwendet wird.

  Im folgenden Beispiel wird gezeigt, wie die erste von drei Java Virtual Machines für einen Katalogservice gestartet wird:

  1. Navigieren Sie wie folgt zum Verzeichnis "bin":

     ```
     cd ObjectGrid-Stammverzeichnis/bin
     ```

  2. Führen Sie den Befehl startOgServer aus:

     ```
     startOgServer.bat|sh cs1 -catalogServiceEndpoints
     cs1:MyServer1.company.com:6601:6602,
     cs2:MyServer2.company.com:6601:6602,
     cs3:MyServer3.company.com:6601:6602
     ```

     In diesem Beispiel wird der Server `cs1` auf dem Host `MyServer1.company.com` gestartet. Dieser Servername ist das erste Argument, das an das Script übergeben wird. Während der Initialisierung von Server `cs1` werden die catalogServiceEndpoints-Parameter untersucht, um die für diesen Prozess zugeord-

neten Ports zu bestimmen. Die Liste wird auch verwendet, um Server cs1 das Annehmen von Verbindungen von anderen Servern (cs2 und cs3) zu ermöglichen.

3. Zum Starten der verbleibenden Katalogserver in der Liste, übergeben Sie die folgenden Argumente an das Script "startOgServer". Es soll der Server cs2 auf dem Host MyServer2.company.com gestartet werden:

```
startOgServer.bat|sh cs2 -catalogServiceEndpoints
cs1:MyServer1.company.com:6601:6602,
cs2:MyServer2.company.com:6601:6602,
cs3:MyServer3.company.com:6601:6602
```

Es soll der Server cs3 auf dem Host MyServer3.company.com gestartet werden:

```
startOgServer.bat|sh cs3 -catalogServiceEndpoints
cs1:MyServer1.company.com:6601:6602,
cs2:MyServer2.company.com:6601:6602,
cs3:MyServer3.company.com:6601:6602
```

Wichtig: Starten Sie alle Katalogserver parallel.

Sie müssen Katalogserver, die in einem Grid enthalten sind, parallel starten, weil jeder Server wartet, bis die anderen Katalogserver der Stammgruppe beitreten. Ein für ein Grid konfigurierter Katalogserver wird erst gestartet, wenn er die anderen Member in der Gruppe identifiziert. Der Katalogserver überschreitet das zulässige Zeitlimit, wenn keine anderen Server verfügbar werden.

- **ORB an eine bestimmte Host/Port-Kombination binden**

  Neben den Ports, die mit dem Argument **catalogServiceEndpoints** definiert werden, verwendet jeder Katalogservice einen Object Request Broker (ORB), um Verbindungen von Clients und Containern zu akzeptieren. Standardmäßig ist der ORB an Port 2809 des lokalen Hosts empfangsbereit. Wenn Sie den ORB an eine bestimmte Host/Port-Kombination in der JVM des Katalogservice binden möchten, verwenden Sie dazu die Argumente **-listenerHost** und **-listenerPort**. Im folgenden Beispiel wird gezeigt, wie Sie einen Katalogserver mit einer einzelnen JVM starten und den zugehörigen ORB an Port 7000 auf MyServer1.company.com binden:

```
startOgServer.sh catalogServer -listenerHost MyServer1.company.com
-listenerPort 7000
```

Jedem eXtreme-Scale-Container und -Client müssen die ORB-Endpunktdaten des Katalogservice bereitgestellt werden. Clients benötigen nur einen Teil dieser Daten, aber Sie sollten für eine hohe Verfügbarkeit mindestens zwei Endpunkte verwenden.

- **Domäne benennen**

  Ein Domänenname ist nicht erforderlich, wenn Sie einen Katalogservice starten. Der Standarddomänenname ist defaultDomain. Wenn Sie Ihre Domäne benennen möchten, verwenden Sie die Option **-domain**. Im folgenden Beispiel wird demonstriert, wie Sie einen Katalogservice mit einer einzelnen JVM und dem Domänennamen myDomain starten.

```
startOgServer.sh catalogServer -domain myDomain
```

- **Sicheren Katalogservice starten**

  Sie können einen sicheren Katalogservice starten, indem Sie die folgenden Argumente angeben:

  -clusterSecurityFile und -clusterSecurityUrl
  Diese Argumente geben die Datei objectGridSecurity.xml an, die die Sicherheitseigenschaften beschreibt, die für alle Server (einschließlich

Katalogservern und Containerservern) gelten. Ein Beispiel für eine solche Eigenschaft ist die Authentifikatorkonfiguration, die die Benutzer-Registry und das Authentifizierungsverfahren darstellt.

-serverProps
Gibt die Servereigenschaftendatei an, die die serverspezifischen Sicherheitseigenschaften enthält. Der Dateiname für diese Eigenschaft kann im herkömmlichen Dateipfadformat angegeben werden, z. B. `c:/tmp/og/catalogserver.props`.

Ein Beispiel zum Starten eines sicheren Katalogservice finden Sie in Schritt 2 des Lernprogramms zur Java-SE-Sicherheit im Handbuch *Produktübersicht*. Ein Beispiel für die Datei `objectGridSecurity.xml` finden Sie im Abschnitt „XML-Sicherheitsdeskriptordatei" auf Seite 192.

Zugehörige Konzepte

„Sichere eXtreme-Scale-Server starten und stoppen" auf Seite 327
Servers müssen für die Implementierungsumgebung häufig sicher sein, und dies erfordert eine spezielle Konfiguration.

Zugehörige Verweise

„Script "startOgServer"" auf Seite 235
Das Script "startOgServer" startet Server. Sie können beim Starten Ihrer Server eine Vielzahl von Parametern verwenden, um die Trace-Erstellung zu aktivieren, Portnummern anzugeben usw.

„Protokolle und Trace" auf Seite 303
Sie können Protokolle und Trace verwenden, um Ihre Umgebung zu überwachen und Fehler zu beheben. Protokolle werden je nach Konfiguration an unterschiedlichen Positionen gespeichert. Möglicherweise müssen Sie einen Trace für einen Server bereitstellen, wenn Sie mit der IBM Unterstützungsfunktion zusammenarbeiten.

Containerprozesse starten

Sie können eXtreme Scale über die Befehlszeile, über eine Implementierungstopologie oder über eine Datei `server.properties` starten.

Warum und wann dieser Vorgang ausgeführt wird

Zum Starten eines Containerprozesses benötigen Sie eine ObjectGrid-XML-Datei. Die ObjectGrid-XML-Datei gibt an, welche eXtreme-Scale-Server im Container enthalten sind. Stellen Sie sicher, dass Ihr Container jedes ObjectGrid in der XML aufnehmen kann, die Sie übergeben. Alle Klassen, die diese ObjectGrids voraussetzen, müssen im Klassenpfad des Containers enthalten sein. Weitere Informationen zur ObjectGrid-XML-Datei finden Sie im Abschnitt „Datei `objectGrid.xsd`" auf Seite 174.

- **Starten Sie den Containerprozess über die Befehlszeile.**
  1. Navigieren Sie über die Befehlszeile zum Verzeichnis "bin":
     ```
     cd ObjectGrid-Stammverzeichnis/bin
     ```
  2. Führen Sie den folgenden Befehl aus:
     ```
     startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

  Wichtig: Im Container wird die Option **-catalogServiceEndPoints** verwendet, um auf den ORB-Host (Object Request Broker) und -Port des Katalogservice zu verweisen. Der Katalogservice verwendet die Optionen **-listenerHost** und **-listenerPort**, um den Host und den Port für die ORB-Bindung anzugeben, oder er

akzeptiert die Standardbindung. Wenn Sie einen Container starten, verwenden Sie die Option **-catalogServiceEndPoints**, um die Werte zu referenzieren, die an die Optionen **-listenerHost** und **-listenerPort** im Katalogservice übergeben werden. Wenn die Optionen **-listenerHost** und **-listenerPort** beim Starten des Katalogservice nicht verwendet werden, stellt der ORB eine Bindung zu Port 2809 auf dem lokalen Host für den Katalogservice her. Verwenden Sie die Option **-catalogServiceEndPoints** nicht, um die Hosts und Ports zu referenzieren, die an die Option **-catalogServiceEndPoints** im Katalogservice übergeben wurden. Im Katalogservice wird die Option **-catalogServiceEndPoints** verwendet, um die erforderlichen Ports für die statische Serverkonfiguration anzugeben. Dieser Prozess wird mit c0 identifiziert, dem ersten Argument, das an das Script übergeben wird. Verwenden Sie die Datei `companyGrid.xml`, um den Container zu starten. Wenn Ihr Katalogserver-ORB auf einem anderen Host ausgeführt wird als Ihr Container oder wenn er einen vom Standard abweichenden Port verwendet, müssen Sie das Argument **-catalogServiceEndPoints** verwenden, um die Verbindung zum ORB herzustellen. Für dieses Beispiel wird angenommen, dass ein einzelner Katalogservice an Port `2809` auf `MyServer1.company.com` aktiv ist.

- **Starten Sie den Container über eine Implementierungsrichtlinie.**

  Obwohl dies nicht erforderlich ist, wird eine Implementierungsrichtlinie für den Containerstart empfohlen. Die Implementierungsrichtlinie wird verwendet, um die Partitionierung und Replikation für eXtreme Scale zu konfigurieren. Die Implementierungsrichtlinie kann auch verwendet werden, um das Verteilungsverhalten zu beeinflussen. Da im vorherigen Beispiel keine Implementierungsrichtliniendatei angegeben wurde, empfängt das Beispiel alle Standardwerte für Replikation, Partitionierung und Verteilung. Deshalb sind die Maps im CompanyGrid in einem einzigen MapSet enthalten. Das MapSet wird nicht partitioniert oder repliziert. Weitere Informationen zu Implementierungsrichtliniendateien finden Sie im Abschnitt „XML-Deskriptordatei für Implementierungsrichtlinie” auf Seite 151. Im folgenden Beispiel wird die Datei `companyGridDpReplication.xml` verwendet, um eine Container-JVM, die JVM ″c0″, zu starten:

  1. Navigieren Sie über die Befehlszeile zum Verzeichnis ″bin″:

     ```
     cd ObjectGrid-Stammverzeichnis/bin
     ```

  2. Führen Sie den folgenden Befehl aus:

     ```
     startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml
     -deploymentPolicyFile ../xml/companyGridDpReplication.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

  Anmerkung: Wenn Sie Java-Klassen haben, die in einem bestimmten Verzeichnis gespeichert sind, können Sie den Server wie folgt mit Argumenten starten, anstatt das Script ″StartOgServer″ zu ändern: `-jvmArgs -cp C:\ . . . \DirectoryPOJOs\POJOs.jar`
  . In der Datei `companyGridDpReplication.xml` enthält ein einzelnes MapSet alle Maps. Dieses MapSet wird in 10 Partitionen aufgeteilt. Jede Partition hat ein synchrones Replikat und keine asynchronen Replikate. Jeder Container, der die Implementierungsrichtlinie `companyGridDpReplication.xml` in Kombination mit der ObjectGrid-XML-Datei `companyGrid.xml` verwendet, kann CompanyGrid-Shards aufnehmen. Starten Sie eine weitere Container-JVM, die JVM ″c1″:

  1. Navigieren Sie über die Befehlszeile zum Verzeichnis ″bin″:

     ```
     cd ObjectGrid-Stammverzeichnis/bin
     ```

  2. Führen Sie den folgenden Befehl aus:

     ```
     startOgServer.sh c1 -objectGridFile ../xml/companyGrid.xml
     -deploymentPolicyFile ../xml/companyGridDpReplication.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

Jede Implementierungsrichtlinie enthält ein oder mehrere Elemente ″objectgrid-Deployment″. Wenn ein Container gestartet wird, veröffentlicht er seine Implementierungsrichtlinie im Katalogservice. Der Katalogservice untersucht jedes Element ″objectgridDeployment″. Wenn der Wert des Attributs ″objectgrid-Name″ mit dem Wert des Attributs ″objectgridName″ eines zuvor empfangenen Elements ″objectgridDeployment″ übereinstimmt, wird das letzte Element ″objectgridDeployment″ ignoriert. Das erste Element ″objectgridDeployment″, das für ein bestimmtes Attribut ″objectgridName″ empfangen wird, wird als Master verwendet. Angenommen, die JVM ″c2″ verwendet eine Implementierungsrichtlinie, die das MapSet in eine andere Anzahl von Partitionen aufteilt:

companyGridDpReplicationModified.xml

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

     <objectgridDeployment objectgridName="CompanyGrid">
        <mapSet name="mapSet1" numberOfPartitions="5"
               minSyncReplicas="1" maxSyncReplicas="1"
            maxAsyncReplicas="0">
            <map ref="Customer" />
            <map ref="Item" />
            <map ref="OrderLine" />
            <map ref="Order" />
        </mapSet>
    </objectgridDeployment>

</deploymentPolicy>
```

Jetzt können Sie eine dritte JVM, die JVM ″c2″, starten.

1. Navigieren Sie über die Befehlszeile zum Verzeichnis ″bin″:

   `cd` *`ObjectGrid-Stammverzeichnis`*`/bin`

2. Führen Sie den folgenden Befehl aus:

```
startOgServer.sh c2 -objectGridFile ../xml/companyGrid.xml
-deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml
-catalogServiceEndPoints MyServer1.company.com:2809
```

Der Container in der JVM ″c2″ wird mit einer Implementierungsrichtlinie gestartet, die 5 Partitionen für mapSet1 angibt. Der Katalogservice enthält jedoch bereits die Masterkopie des objectgridDeployment-Objekt für das CompanyGrid. Als die JVM ″c0″ gestartet wurde, waren 10 Partitionen für dieses MapSet vorhanden. Weil dies der erste Container war, der gestartet wurde und seine Implementierungsrichtlinie veröffentlicht hat, wurde seine Implementierungsrichtlinie als Master definiert. Deshalb wird jeder objectgridDeployment-Attributwert, der `CompanyGrid` entspricht, in einer nachfolgenden Implementierungsrichtlinie ignoriert.

- **Starten Sie einen Container über eine Servereigenschaftendatei.**

  Sie können eine Servereigenschaftendatei verwenden, um die Trace-Erstellung zu und die Sicherheit für einen Container zu konfigurieren. Führen Sie die folgenden Befehle aus, um Container ″c3″ über eine Servereigenschaftendatei zu starten:

  1. Navigieren Sie über die Befehlszeile zum Verzeichnis ″bin″:

     `cd` *`ObjectGrid-Stammverzeichnis`*`/bin`

  2. Führen Sie den folgenden Befehl aus:

```
startOgServer.sh c3 -objectGridFile ../xml/companyGrid.xml
-deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml
-catalogServiceEndPoints MyServer1.company.com:2809
-serverProps ../serverProps/server.properties
```

  Im Folgenden sehen Sie eine Beispieldatei `server.properties`:

```
server.properties
workingDirectory=
traceSpec=*=all=disabled
systemStreamToFileEnabled=trueenableMBeans=true
memoryThresholdPercentage=50
```

Dies ist die Basisservereigenschaftendatei, in der die Sicherheit nicht aktiviert ist. Weitere Informationen zur Datei `server.properties` finden Sie im Abschnitt „Servereigenschaftendatei" auf Seite 197.

Zugehörige Konzepte

„Sichere eXtreme-Scale-Server starten und stoppen" auf Seite 327
Servers müssen für die Implementierungsumgebung häufig sicher sein, und dies erfordert eine spezielle Konfiguration.

Zugehörige Verweise

„Script ″startOgServer″"
Das Script ″startOgServer″ startet Server. Sie können beim Starten Ihrer Server eine Vielzahl von Parametern verwenden, um die Trace-Erstellung zu aktivieren, Portnummern anzugeben usw.

„Protokolle und Trace" auf Seite 303
Sie können Protokolle und Trace verwenden, um Ihre Umgebung zu überwachen und Fehler zu beheben. Protokolle werden je nach Konfiguration an unterschiedlichen Positionen gespeichert. Möglicherweise müssen Sie einen Trace für einen Server bereitstellen, wenn Sie mit der IBM Unterstützungsfunktion zusammenarbeiten.

Script ″startOgServer″

Das Script ″startOgServer″ startet Server. Sie können beim Starten Ihrer Server eine Vielzahl von Parametern verwenden, um die Trace-Erstellung zu aktivieren, Portnummern anzugeben usw.

Zweck

Sie können das Script ″startOgServer″ verwenden, um Server zu starten.

Position

Sie finden das Script ″startOgServer″ im Verzeichnis ″bin″ des Stammverzeichnisses, z. B.:

```
cd ObjectGrid-Stammverzeichnis/bin
```

Anmerkung: Wenn Sie Java-Klassen haben, die in einem bestimmten Verzeichnis gespeichert sind, können Sie den Server wie folgt mit Argumenten starten, anstatt das Script ″StartOgServer″ zu ändern: `-jvmArgs -cp C:\ . . . \DirectoryPOJOs\POJOs.jar`

.

Syntax für Katalogserver

Verwenden Sie zum Starten eines Katalogservers die folgenden Befehle:

.

```
startOgServer.bat <Server> [Optionen]
```

.

```
startOgServer.sh <Server>[Optionen]
```

Verwenden Sie zum Starten des konfigurierten Standardkatalogservers die folgenden Befehle:

```
startOgServer.bat catalogServer
```

```
startOgServer.sh catalogServer
```

Optionen für das Starten von Katalogservern

Parameter für das Starten eines Katalogservers:

-catalogServiceEndPoints <Server:Serverhost:Clientport:Peer-Port,Server:Serverhost:Clientport:Peer-Port>
Im Container wird die Option **-catalogServiceEndpoints** verwendet, um auf den ORB-Host (Object Request Broker) und -Port des Katalogservice zu verweisen.

-clusterSecurityFile <XML-Sicherheitsdatei des Clusters>
Gibt die Position der XML-Sicherheitsdeskriptordatei an.

-clusterSecurityUrl <URL der Sicherheits-XML des Clusters>

-domain <Domänenname>

-listenerHost <Hostname>
Standardeinstellung: `localhost`

Gibt den Listener-Host für die Kommunikation mit Internet Inter-ORB Protocol (IIOP) an.

-listenerPort <Port>
Standardeinstellung: `2809`

Gibt den Listener-Port für die Kommunikation mit IIOP an.

-serverProps <Servereigenschaftendatei>
Gibt die Servereigenschaftendatei an, die die serverspezifischen Sicherheitseigenschaften enthält. Der Dateiname für diese Eigenschaft kann im herkömmlichen Dateipfadformat angegeben werden, z. B. `c:/tmp/og/catalogserver.props`.

-JMXServicePort <Port>
Standardeinstellung: `1099`

Gibt die Portnummer für die Kommunikation mit Java Management Extensions (JMX) an.

-traceSpec <Trace-Spezifikation>
Gibt eine Zeichenfolge an, die den Geltungsbereich des Trace angibt, der beim Serverstart aktiviert wird.

Beispiel:

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <Trace-Datei>
Gibt den Pfad der Datei an, in der Trace-Informationen gespeichert werden.

Beispiel: `../logs/c4Trace.log`

-timeout <Sekunden>
Gibt das Zeitlimit (in Sekunden) für den Serverstart an.

-script <Scriptdatei>

-jvmArgs <JVM-Argumente>
Gibt eine Gruppe von JVM-Argumenten an. Jeder Parameter hinter dem Parameter **-jvmArgs** wird verwendet, um die Server-JVM zu starten. Wenn der Parameter **-jvmArgs** verwendet wird, müssen Sie sicherstellen, dass er das letzte optionale Scriptargument ist, das angegeben wird.

Beispiel: `-jvmArgs -Xms256M -Xmx1G`

Syntax für Containerserver

```
startOgServer.bat <Server> -objectgridFile <XML-Datei>
-deploymentPolicyFile <XML-Datei> [Optionen]
```

```
startOgServer.bat <Server> -objectgridUrl <XML-URL>
-deploymentPolicyUrl <XML-URL> [Optionen]
```

```
startOgServer.sh <Server> -objectgridFile <XML-Datei>
-deploymentPolicyFile <XML-Datei> [Optionen]
```

```
 startOgServer.sh <Server> -objectgridUrl <XML-URL>
-deploymentPolicyUrl <XML-URL> [Optionen]
```

Optionen für Containerserver

-catalogServiceEndPoints<Hostname:Port,Hostname:Port>
Standardeinstellung: localhost:2809

-deploymentPolicyFile <XML-Implementierungsrichtliniendatei>
Gibt den Pfad der Implementierungsrichtliniendatei an.

Beispiel: `../xml/SimpleDP.xml`

-deploymentPolicyUrl <URL der XML-Implementierungsrichtliniendatei>

-listenerHost <Hostname>
Standardeinstellung: `localhost`

Gibt den Listener-Host für die Kommunikation mit Internet Inter-ORB Protocol (IIOP) an.

-listenerPort <Port>
Standardeinstellung: `2809`

Gibt den Listener-Port für die Kommunikation mit IIOP an.

-serverProps <Servereigenschaftendatei>
Gibt den Pfad der Servereigenschaftendatei an.

Beispiel: `../security/server.props`

-zone <Zonenname>
Gibt die für alle Container im Server zu verwendende Zone an.

-traceSpec <Trace-Spezifikation>
Gibt eine Zeichenfolge an, die den Geltungsbereich des Trace angibt, der beim Serverstart aktiviert wird.

Beispiel:

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <Trace-Datei>
Gibt den Pfad der Datei an, in der Trace-Informationen gespeichert werden.

Beispiel: `../logs/c4Trace.log`

-timeout <Sekunden>
Gibt das Zeitlimit (in Sekunden) für den Serverstart an.

-script <Scriptdatei>

-jvmArgs <JVM-Argumente>
Gibt eine Gruppe von JVM-Argumenten an. Jeder Parameter hinter dem Parameter **-jvmArgs** wird verwendet, um die Server-JVM zu starten. Wenn der Parameter **-jvmArgs** verwendet wird, müssen Sie sicherstellen, dass er das letzte optionale Scriptargument ist, das angegeben wird.

Beispiel: `-jvmArgs -Xms256M -Xmx1G`

Zugehörige Konzepte

„Sichere eXtreme-Scale-Server starten und stoppen" auf Seite 327
Servers müssen für die Implementierungsumgebung häufig sicher sein, und dies erfordert eine spezielle Konfiguration.

Zugehörige Tasks

„Eigenständige Server von WebSphere eXtreme Scale starten" auf Seite 228
Wenn Sie eine eigenständige Konfiguration von WebSphere eXtreme Scale verwenden, setzt sich die Umgebung aus Katalogservern, Containerservern und eXtreme-Scale-Clientprozessen zusammen. Sie müssen diese Prozesse manuell konfigurieren und starten.

„Katalogservice in einer eigenständigen Umgebung starten" auf Seite 229
Sie müssen den Katalogservice manuell starten, wenn Sie eine verteilte Umgebung von WebSphere eXtreme Scale ohne WebSphere Application Server verwenden.

„Containerprozesse starten" auf Seite 232
Sie können eXtreme Scale über die Befehlszeile, über eine Implementierungstopologie oder über eine Datei `server.properties` starten.

TCP-Ports im eigenständigen Modus konfigurieren

Eine Java Virtual Machine, in der eine Katalogserviceinstanz ausgeführt wird, erfordert vier Ports. Zwei Ports sind für interne Verwendung bestimmt, der dritte Port wird für Clients und Container-Shards für die Kommunikation mit Internet Inter-ORB Protocol (IIOP) verwendet, und der vierte Port wird für die JMX-Kommunikation (Java Management Extensions) verwendet.

Warum und wann dieser Vorgang ausgeführt wird

JVM-Endpunkte für den Katalogservice

eXtreme Scale verwendet IIOP hauptsächlich für die Komunikation zwischen JVMs. Die JVMs des Katalogservice sind die einzigen JVMs, die eine explizite Konfiguration der Ports für die IIOP-Services und der Ports für die Gruppenservices erfordern. Die internen Ports werden mit der Befehlszeilenoption **-catalogServiceEndPoints** angegeben.

`-catalogServiceEndPoints <Server:Host:Port:Port,Server:Host:Port:Port>`

Mit der Befehlszeilenoption **-catalogServiceEndPoints** können Sie zwei Ports pro Server konfigurieren. Die IIOP-Ports werden mit den folgenden Befehlszeilenoptionen konfiguriert:

```
-listenerHost <Hostname>
-listenerPort <Port>
```

Geben Sie beim Starten jeder JVM des Katalogservice die vollständige Gruppe der Katalogserviceendpunkte zusammen mit einem einzigen Listener-Port für diese JVM an.

Endpunkte der Container-JVMs

Die Container-JVMs verwenden zwei Ports. Ein Port ist für die interne Verwendung bestimmt und der andere für die IIOP-Kommunikation. Die Container-JVMs suchen automatisch nicht verwendete Ports und konfigurieren sich anschließend selbst für die Verwendung der beiden dynamisch erstellten Ports. Dieser automatische Prozess minimiert die Konfiguration. Wenn jedoch Firewalls verwendet werden oder wenn Sie Ports explizit konfigurieren möchten, können Sie eine Befehlszeilenoption verwenden, um den zu verwendenden ORB-Port anzugeben:

```
-listenerHost <Hostname>
-listenerPort <Port>
```

Mit diesen Befehlszeilenoptionen können Sie den Hostnamen (wichtig für die Bindung an die richtige Netzkarte) und den Port für diese JVM angeben. Sie können den internen Port mit dem Befehlszeilenparameter **-haManagerPort** angeben. Für die einfachste Konfiguration können Sie die Ports jedoch von der Laufzeitumgebung auswählen lassen.

1. Starten Sie den ersten Katalogserver auf hostA. Im Folgenden sehen Sie einen Beispielbefehl:

   ```
   ./startOgServer.sh cs1 -listenerHost hostA -listenerPort 2809
   -catalogServiceEndPoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
   ```

2. Starten Sie den zweiten Katalogserver auf hostB. Im Folgenden sehen Sie einen Beispielbefehl:

   ```
   ./startOgServer.sh cs2 -listenerHost hostB -listenerPort 2809
   -catalogServiceEndPoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
   ```

   Clients müssen nur die Listener-Endpunkte des Katalogservice kennen. Clients rufen Endpunkte für Container-JVMs, die die JVMs sind, die die Daten enthalten, automatisch vom Katalogservice ab. Um eine Verbindung zu dem Katalogservice aus dem vorherigen Beispiel herzustellen, muss der Client die folgende Liste von `Host:Port`-Paaren an die API "connect" übergeben:

   ```
   hostA:2809,hostB:2809
   ```

   Verwenden Sie zum Starten einer Container-JVM, die den Beispielkatalogservice verwendet, den folgenden Befehl:

   ```
   ./startOgServer.sh c0 -catalogServiceEndPoints hostA:2809,hostB:2809
   ```

Eigenständige eXtreme-Scale-Server stoppen

Sie können das Script `stopOgServer.sh` für UNIX bzw. das Script `stopOgServer.bat` für Windows verwenden, um Serverprozesse zu stoppen.

- Stoppen Sie die eXtreme-Scale-Containerprozesse.
  1. Navigieren Sie über die Befehlszeile zum Verzeichnis "bin".

     ```
     cd ObjectGrid-Stammverzeichnis/bin
     ```

  2. Führen Sie das Script stopOgServer aus, um den Server zu stoppen. Im folgenden Beispiel wird der Server "c0" gestoppt:

```
stopOgServer.sh c0 -catalogServiceEndPoints MyServer1.company.com:2809
```

Achtung: Im Container wird die Option **-catalogServiceEndPoints** verwendet, um auf den ORB-Host (Object Request Broker) und -Port des Katalogservice zu verweisen. Der Katalogservice verwendet die Optionen **-listenerHost** und **-listenerPort**, um den Host und den Port für die ORB-Bindung anzugeben, oder er akzeptiert die Standardbindung. Wenn Sie einen Container stoppen, verwenden Sie die Option **-catalogServiceEndPoints**, um die Werte zu referenzieren, die an die Optionen **-listenerHost** und **-listenerPort** im Katalogservice übergeben werden. Wenn die Optionen **-listenerHost** und **-listenerPort** beim Starten des Katalogservice nicht verwendet werden, stellt der ORB eine Bindung zu Port 2809 auf dem lokalen Host für den Katalogservice her.

Verwenden Sie die Option **-catalogServiceEndPoints** nicht, um die Hosts und Ports zu referenzieren, die an die Option **-catalogServiceEndPoints** im Katalogservice übergeben wurden. Verwenden Sie im Katalogservice die Option **-catalogServiceEndPoints**, um die Ports anzugeben, die für die Konfiguration eines statischen Servers erforderlich sind.

- Stoppen Sie die eXtreme-Scale-Katalogserviceprozesse.
  1. Navigieren Sie über die Befehlszeile zum Verzeichnis "bin".
     ```
     cd ObjectGrid-Stammverzeichnis/bin
     ```
  2. Führen Sie das Script stopOgServer aus, um den Server zu stoppen.
     ```
     stopOgServer.sh catalogServer -bootstrap MyServer1.company.com:6601
     ```

  Geben Sie den zu stoppenden Prozess mit dem Argument "catalogServer" an. Dies ist das erste Argument, das an das Script übergeben wird. Für dieses Beispiel wird angenommen, dass der Katalogservice in der Domäne "MyServer1.company.com" mit dem clientAccessPort-Wert 6601 gestartet wurde.
- Aktivieren Sie die Trace-Erstellung für den Prozess zum Stoppen des Servers. Wenn ein Container nicht gestoppt werden kann, können Sie die Trace-Erstellung als Unterstützung für die Fehlerbehebung aktivieren. Zum Aktivieren der Trace-Erstellung beim Stoppen eines Servers fügen Sie den Stoppbefehlen die Parameter **-traceSpec** und **-traceFile** hinzu. Der Parameter **-traceSpec** gibt den Typ des zu aktivierenden Trace an, und der Parameter **-traceFile** gibt den Pfad und den Namen der für die Trace-Daten zu erstellenden und zu verwendenden Datei an.
  1. Navigieren Sie über die Befehlszeile zum Verzeichnis "bin".
     ```
     cd ObjectGrid-Stammverzeichnis/bin
     ```
  2. Führen Sie das Script stopOgServer mit aktivierter Trace-Erstellung aus.
     ```
     stopOgServer.sh c4 -catalogServiceEndPoints MyServer1.company.com:2809
      -traceFile ../logs/c4Trace.log -traceSpec ObjectGrid=all=enabled
     ```

     Suchen Sie nach der Trace-Erstellung nach Fehlern, die sich auf Portkonflikte, fehlende Klassen, fehlende oder ungültige XML-Dateien oder Stack-Traces beziehen. Empfohlene Trace-Spezifikation für den Start sind:
     - ObjectGrid=all=enabled
     - ObjectGrid*=all=enabled

     Informationen zu allen Optionen für die Trace-Spezifikation finden Sie im Abschnitt „Trace-Optionen” auf Seite 306.

Zugehörige Verweise

„Script ″stopOgServer″″
Das Script ″stopOgServer″ stoppt Server.

Script ″stopOgServer″

Das Script ″stopOgServer″ stoppt Server.

Zweck

Sie können das Script ″stopOgServer″ verwenden, um Server zu stoppen.

Position

Sie finden das Script ″stopOgServer″ im Verzeichnis ″bin″ des Stammverzeichnisses, z. B.:

```
cd ObjectGrid-Stammverzeichnis/bin
```

Syntax

Verwenden Sie zum Stoppen eines Katalogservers die folgenden Befehle:

```
stopOgServer.bat <Server> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [Optionen]
```

```
stopOgServer.sh <server> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [Optionen]
```

Verwenden Sie zum Stoppen eines ObjectGrid-Containerservers die folgenden Befehle:

```
stopOgServer.bat <Server> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [Optionen]
```

```
startOgServer.sh -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [Optionen]
```

Optionen

-clientSecurityFile <Sicherheitseigenschaftendatei>

-traceSpec <Trace-Spezifikation>
Gibt eine Zeichenfolge an, die den Geltungsbereich des Trace angibt, der beim Serverstart aktiviert wird.

Beispiel:

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <Trace-Datei>
Gibt den Pfad der Datei an, in der Trace-Informationen gespeichert werden.

Beispiel: `../logs/c4Trace.log`

-jvmArgs <JVM-Argumente>
Jeder Parameter hinter der Option ″-jvmArgs″ wird verwendet, um die Server-

JVM zu starten. Wenn die Option "-jvmArgs" verwendet wird, müssen Sie sicherstellen, dass er das letzte optionale Scriptargument ist, das angegeben wird.

Zugehörige Tasks

„Eigenständige eXtreme-Scale-Server stoppen" auf Seite 239
Sie können das Script `stopOgServer.sh` für UNIX bzw. das Script `stopOgServer.bat` für Windows verwenden, um Serverprozesse zu stoppen.

WebSphere eXtreme Scale mit WebSphere Application Server verwalten

Sie können Katalogservice- und Containerserverprozesse in WebSphere Application Server ausführen. Der Prozess zum Konfigurieren dieser Server unterscheidet sich von dem in einer eigenständigen Konfiguration. Der Katalogservice kann automatisch in Servern oder im Deployment Manager von WebSphere Application Server gestartet werden. Der Containerprozess wird gestartet, wenn eine eXtreme-Scale-Anwendung in der Umgebung von WebSphere Application Server implementiert und gestartet wird.

Zugehörige Verweise

„Servereigenschaftendatei" auf Seite 197
Die Servereigenschaftendatei enthält verschiedene Eigenschaften, mit denen die verschiedenen Einstellungen für Ihren Server definiert werden, z. B. Trace-Einstellungen, Protokollierung und Sicherheitskonfiguration. Die Servereigenschaftendatei wird vom Katalogservice und von den Containerservern verwendet.

„Clienteigenschaftendatei" auf Seite 203
Sie können eine Eigenschaftendatei erstellen, die auf Ihren Anforderungen für eXtreme-Scale-Clientprozesse basiert.

„Referenz der Eigenschaftendatei" auf Seite 196
Servereigenschaftendateien enthalten Einstellungen für die Ausführung der Katalogserver und Containerserver. Sie können eine Servereigenschaftendatei für eine eigenständige Konfiguration oder eine Konfiguration von WebSphere Application Server angeben. Clienteigenschaftendateien enthalten Einstellungen für Ihren Client.

Katalogserviceprozess in einer Umgebung mit WebSphere Application Server starten

Ein einzelner eXtreme-Scale-Katalogservice kann in einer Umgebung mit WebSphere Application Server oder WebSphere Application Server Network Deployment automatisch gestartet werden. Sie können den Katalogservice so konfigurieren, dass er in einem beliebigen Prozess in der WebSphere-Zelle ausgeführt werden kann. Ein einzelner Katalogservice ohne Cluster ist für Entwicklungsumgebungen akzeptabel. Für eine Produktionsumgebung sollten Sie ein Katalogservice-Grid verwenden.

Vorbereitungen

Sie müssen WebSphere Application Server und WebSphere eXtreme Scale installieren. Weitere Informationen finden Sie im Abschnitt Kapitel 4, „WebSphere eXtreme Scale installieren und implementieren", auf Seite 27.

Warum und wann dieser Vorgang ausgeführt wird

Der Katalogserviceprozess wird in Prozessen von WebSphere Application Server ausgeführt. Wenn der Prozess, der den Katalogserviceprozess enthält, gestartet wird, wird auch der Katalogserviceprozess gestartet. Im Basisprodukt WebSphere Application Server wird der Katalogservice im Serverprozess ausgeführt. In WebSphere Application Server Network Deployment wird der Katalogserviceprozess automatisch im Deployment Manager ausgeführt, aber Sie können den Katalogservice auch so konfigurieren, dass er in einem Node-Agent- oder Anwendungsserverprozess ausgeführt wird.

- **Einzelnen Katalogserver im Basisprodukt WebSphere Application Server starten**

  Wenn WebSphere eXtreme Scale in WebSphere Application Server installiert ist, wird der Katalogservice automatisch im Serverprozess gestartet. Dieses Feature vereinfacht die Komponententests in Entwicklungsumgebungen wie Rational Application Developer, weil Sie den Katalogservice nicht explizit starten müssen.

  Wenn WebSphere eXtreme Scale in WebSphere Application Server Network Deployment installiert ist, wird der Katalogservice automatisch im Deployment-Manager-Prozess gestartet.

- **Katalogservice-Grid in WebSphere Application Server Network Deployment starten**

  Das Katalogserver-Grid von eXtreme Scale kann über die angepasste Eigenschaft ″catalog.services.cluster″, die für die Zelle von WebSphere Application Server gesetzt wird, explizit in einem Deployment-Manager-, Node-Agent- oder Anwendungsserverprozess definiert werden. Definieren Sie den Wert für die angepasste Eigenschaft im folgenden Format: *Servername*:*Hostname*:*Clientport*-:Peer-Port:*Listener-Port*. Sie können mehrere Server angeben, indem Sie die einzelnen Werte durch Kommas trennen.

  Führen Sie Katalogservices in einer Produktionsumgebung nicht in demselben Prozess aus wie eXtreme-Scale-Container. Führen Sie den Katalogservice in mehreren Node-Agent-Prozessen oder in einem Anwendungsserver-Grid ohne eine eXtreme-Scale-Anwendung aus. Mit einigen Ausnahmen entspricht die angepasste Eigenschaft ″catalog.services.cluster″ dem Parameter **-catalogServiceEndPoints**, der beim Starten eines eigenständigen Katalogservers zusammen mit dem Parameter **-listenerPort** verwendet wird. Jedes Attribut für die angepasste Eigenschaft ist wie folgt definiert:

  serverName
  Gibt den vollständig qualifizierten Namen des WebSphere-Prozesses, z. B. cellName, nodeName oder serverName, des Servers an, in dem der Katalogservice ausgeführt wird.

  Beispiel: `cellA\node1\nodeagent`

  hostName
  Gibt den Namen des Hostservers an.

  clientPort
  Gibt den Port an, der für die Peer-Kommunikation im Katalog-Grid verwendet wird.

  peerPort
  Gibt den Port an, der für die Peer-Kommunikation im Katalog-Grid verwendet wird.

listenerPort
Der listenerPort-Wert muss mit dem BOOTSTRAP_ADDRESS-Wert übereinstimmen, der in der Konfiguration des WebSphere-Servers definiert ist.

Zum Erstellen der angepassten Eigenschaft klicken Sie in der Administrationskonsole auf **Systemverwaltung → Zelle → Angepasste Eigenschaften → Neu**. Geben Sie `catalog.services.cluster` als Namen und den Wert im entsprechenden Format über die definierten Attribute an. Es folgt ein Beispiel für einen gültigen Wert:

```
cellA\node1\nodeagent:host.local.domain:6600:6601:2809,cellA\node2\
nodeagent:host.foreign.domain:6600:6601:2809
```

Nachdem Sie die angepasste Eigenschaft definiert haben, müssen Sie jeden angegebenen Server erneut starten. Wenn Sie diese Server erneut starten, wird das Katalogservice-Grid automatisch gestartet.

Einschränkung: Das Katalog-Grid verwendet die Stammgruppenmechanismen von WebSphere Application Server. Deshalb sind die Member eines Katalog-Grids auf eine Stammgruppe beschränkt, und deshalb ist das Katalog-Grid auf eine Zelle beschränkt. eXtreme Scale kann die Zellengrenzen jedoch überwinden, indem eine Verbindung zu einem Katalogserver über die Zellengrenzen hinweg hergestellt wird, z. B. zu einem eigenständigen Katalog-Grid oder zu einem Katalog-Grid in einer anderen Zelle.

Containerprozesse in einer Umgebung mit WebSphere Application Server starten

Containerprozesse in einer Umgebung mit WebSphere Application Server werden automatisch gestartet, wenn ein Modul gestartet wird, in dem die XML-Dateien von eXtreme Scale enthalten sind.

Vorbereitungen

WebSphere Application Server und WebSphere eXtreme Scale müssen installiert und in der Lage sein, auf die Administrationskonsole von WebSphere Application Server zuzugreifen.

Warum und wann dieser Vorgang ausgeführt wird

Java-EE-Anwendungen haben komplexe Klassenladeregeln, die das Laden von Klassen erheblich komplexer machen, wenn eine gemeinsame Konfiguration von WebSphere eXtreme Scale in einem Java-EE-Server verwendet wird. Eine Java-EE-Anwendung ist gewöhnlich eine einzelne EAR-Datei, in der eine oder mehrere EJB- oder WAR-Module implementiert sind.

WebSphere eXtreme Scale überwacht den Start jedes Moduls und sucht nach XML-Dateien von eXtreme Scale. Wenn WebSphere eXtreme Scale feststellt, dass ein Modul mit den XML-Dateien gestartet wird, wird der Anwendungsserver als eXtreme-Scale-Container-JVM beim Katalogservice registriert. Durch die Registrierung der Container beim Katalogservice kann dieselbe Anwendung in mehreren Grids implementiert und trotzdem vom Katalogservice als einzelnes Grid behandelt werden. Der Katalogservice kümmert sich nicht um Zellen, Grids oder dynamische Grids. Die einzige Unterscheidung, die der Katalogservice trifft, ist die, ob eine Komponente eine eXtreme-Scale-Container-JVM ist oder nicht. Ein einzelnes

logisches Grid kann sich bei Bedarf über mehrere Zellen von WebSphere Extended Deployment erstrecken.

1. Packen Sie Module in Ihre EAR-Datei, die die XML-Dateien für eXtreme Scale im Ordner META-INF enthalten. WebSphere eXtreme Scale erkennt das Vorhandensein der Dateien `objectGrid.xml` und `objectGridDeployment.xml` im Ordner META-INF von EJB- und WEB-Modulen, wenn diese gestartet werden. Wenn nur eine Datei `objectGrid.xml` gefunden wird, wird davon ausgegangen, dass die JVM ein Client ist. Sind beide Dateien vorhanden, wird davon ausgegangen, dass diese JVM als Container für das Grid agiert, das in der Datei `objectGridDeployment.xml` definiert ist.

   Sie müssen die richtigen Namen für diese XML-Dateien verwenden. Bei den Dateinamen muss die Groß-/Kleinschreibung beachtet werden. Wenn die Dateien nicht vorhanden sind, wird der Container nicht gestartet. Sie können in der Datei `systemout.log` nach Nachrichten suchen, die darauf hinweisen, dass Shards verteilt werden. Ein EJB-Modul oder WAR-Modul, das eXtreme Scale verwendet, muss XML-Dateien von eXtreme Scale in seinem Verzeichnis `META-INF` enthalten.

   Die XML-Dateien von eXtreme Scale enthalten Folgendes:

   - eine Datei `objectGrid.xml`, die allgemein auch als XML-Deskriptordatei von eXtreme Scale bezeichnet wird,
   - eine Datei `objectGridDeployment.xml`, die allgemein auch als XML-Implementierungsdeskriptordatei bezeichnet wird,
   - eine Datei `entity.xml`, falls Entitäten verwendet werden (optional). Der Name der Datei `entity.xml` muss mit dem Namen übereinstimmen, der in der Datei `objectGrid.xml` angegeben ist.

   Die Laufzeitumgebung von eXtreme Scale erkennt diese Dateien und stellt dann eine Verbindung zum Katalogservice her, um ihn darüber zu informieren, dass ein weiterer Container verfügbar ist, der Shards für diese Instanz von eXtreme Scale aufnehmen kann.

   Tipp: Wenn Sie eine Anwendung mit Entitäten verwenden möchten, die einen einzigen eXtreme-Scale-Server verwenden, setzen Sie minSyncReplicas in der XML-Implementierungsdeskriptordatei auf `0`, um zu verhindern, dass die Nachricht `CWPRJ1005E: Fehler beim Auflösen der Entitätszuordnung. Ausnahme: The EntityMetadata repository is not available. Timeout threshold reached` ausgegeben wird.
2. Implementieren und starten Sie Ihre Anwendung.

   Der Container wird automatisch gestartet, wenn das Modul gestartet wird. Der Katalogservice beginnt sobald wie möglich mit der Verteilung der primären und Replikat-Shards der Partition. Diese Verteilung findet sofort statt, sofern Sie kein Attribut ″numInitialContainers″ in der Datei `objectGridDeployment.xml` definieren. Wenn Sie das Attribut ″numInitialContainers″ definieren, findet die Verteilung statt, wenn die angegebene Anzahl an Container gestartet wurde.

Nächste Maßnahme

Anwendungen, die sich in derselben Zelle wie die Container befinden, können eine Verbindung zu diesen Grid über die Methode ″ObjectGridManager.connect(null, null) herstellen und anschließend die Methode ″getObjectGrid(ccc, ″object grid name″)″ aufrufen. Die Methoden ″connect″ und ″getObjectGrid″ können blockiert werden, bis die Container die Shards verteilt haben, aber diese Blockierung tritt nur auf, wenn das Grid gestartet wird.

ClassLoaders

Alle Plug-ins oder Objekte, die in eXtreme Scale gespeichert sind, werden in ein bestimmtes Klassenladeprogramm geladen. Zwei EJB-Module in derselben EAR-Datei können diese Objekte enthalten. Die Objekte sind identisch, werden aber von verschiedenen Klassenladeprogrammen geladen. Wenn Anwendung A ein Person-Objekt in einer eXtreme-Scale-Map, die eine lokale Map des Server ist, empfängt Anwendung B eine Ausnahme des Typs ″ClassCastException″, wenn sie versucht, das Objekt zu lesen. Diese Ausnahme tritt ein, weil Anwendung B das Person-Objekt in einem anderen Klassenladeprogramm geladen hat.

Eine Methode zur Behebung dieses Problems ist die Verwendung eines Stammmoduls, das die erforderlichen Plug-ins und Objekte enthält, die in eXtreme Scale gespeichert werden. Jedes Modul, das eXtreme Scale verwendet, muss dieses Modul für seine Klassen referenzieren. Eine andere Lösung ist die Verwaltung dieser gemeinsam genutzten Objekte in einer Dienstprogramm-JAR-Datei, die in einem Klassenladeprogramm enthalten ist, das von Modulen und Anwendungen gemeinsam genutzt wird. Die Objekte können auch in WebSphere-Klassen oder in ein Verzeichnis `lib/ext` gestellt werden, aber dies macht die Implementierung komplexer und wird deshalb nicht empfohlen.

EJB-Module in einer EAR-Datei nutzen gewöhnlich dasselbe Klassenladeprogramm und sind von diesem Problem nicht betroffen. Jedes WAR-Modul hat ein eigenes Klassenladeprogramm und ist von diesem Problem betroffen.

Verbindung zu einem Grid als reiner Client herstellen

Wenn die Eigenschaft ″catalog.services.cluster″ in den angepassten Eigenschaften für die Zelle, den Knoten oder den Server definiert ist, kann jedes Modul in der EAR-Datei die Methode ″ObjectGridManager.connect (ServerFactory.getServerProperties().getCatalogServiceBootstrap(), null, null)″ aufrufen, um ein ClientClusterContext-Objekt abzurufen, und die Methode ″ObjectGridManager.getObjectGrid(ccc, ″grid name″)″, um eine Referenz auf eXtreme Scale abzurufen. Wenn Anwendungsobjekte in Maps gespeichert werden, müssen Sie sicherstellen, dass diese Objekte in einem gemeinsamen Klassenladeprogramm vorhanden sind.

Java-SE-Clients und Clients außerhalb der Zelle können über den Bootstrap-IIOP-Port des Katalogservice (standardmäßig der Deployment Manager in WebSphere) ein ClientClusterContext-Objekt und anschließend wie gewöhnlich eine benannte Instanz von eXtreme Scale abrufen.

EntityManager

Der EntityManager hilft, weil die Tupel in den Maps und nicht die Anwendungsobjekte werden, damit das Klassenladeprogramm weniger Probleme hat, um die es sich kümmern muss. Plug-ins können jedoch ein Problem sein. Beachten Sie auch, das immer eine eXtreme-Scale-XML-Deskriptordatei für Clientkorrekturwerte erforderlich ist, wenn eine Verbindung zu einer Instanz von eXtreme Scale hergestellt wird, in der Entitäten definiert sind: ObjectGridManager.connect(″host-:port[,host:port], null, objectGridOverride) oder ObjectGridManager.connect(null, objectGridOverride).

TCP-Ports (Transmission Control Protocol) in einer Umgebung mit WebSphere Application Server konfigurieren

Der Katalogservice wird standardmäßig in einer einzelnen Instanz im Deployment Manager ausgeführt und verwendet den IIOP-Bootstrap-Port (Internet Inter-ORB Protocol) für die Java Virtual Machine des Deployment Manager.

Warum und wann dieser Vorgang ausgeführt wird

Webanwendungen und EJB-Anwendungen in der Zelle können mit dem Connect-Aufruf `null,null` eine Verbindung zu Grids herstellen, die sich in derselben Zelle befinden, anstatt Bootstrap-Ports für den Katalogservice anzugeben.

Wenn das Katalogservice-Grid in WebSphere Application Server vom Deployment Manager verwaltet wird, müssen Clients außerhalb der Zelle (einschließlich Java-SE-Clients) mit dem Hostnamen des Deployment Manager und dem IIOP-Bootstrap-Port eine Verbindung zum Katalogservice herstellen.

Wenn der Katalogservice in Zellen von WebSphere Application Server ausgeführt wird, während sich die Clients außerhalb der Zellen befinden, müssen Sie Clientverbindungsports über die angepassten Zelleneigenschaften mit dem Namen `catalog.services.cluster` suchen. Wenn Sie diesen Eintrag finden, verwenden Sie ihn, um die Clients mit dem Katalogservice zu verbinden. Wenn Sie den Eintrag nicht finden, verwenden Sie den IIOP-Port des Deployment Manager für die Clientverbindung. Befinden sich Ihre Clients in Zellen von WebSphere Application Server, können Sie Ports direkt von der Schnittstelle ″CatalogServerProperties″ abrufen.

In eXtreme Scale werden die DCS-Ports (Distribution and Consistency Services) des High Availability Manager für die Gruppenzugehörigkeit wiederverwendet. Auch die JMX-Ports (Java Management Extensions) werden wiederverwendet.

Verwaltung von HTTP-Sitzungen

In einer Umgebung mit WebSphere Application Server Version 5 oder höher kann der mit WebSphere eXtreme Scale bereitgestellte HTTP-Sitzungsmanager den Standardsitzungsmanager im Anwendungsserver außer Kraft setzen.

Der HTTP-Sitzungsmanager von WebSphere eXtreme Scale kann auch in WebSphere Application Server Version 6.0.2 oder höher oder auch in einer Umgebung ohne WebSphere Application Server, wie z. B. WebSphere Application Server Community Edition oder Apache Tomcat ausgeführt werden.

Features

Der Sitzungsmanager wurde so konzipiert, dass er in jedem Container der Java Platform, Enterprise Edition Version 1.4 ausgeführt werden kann. Der Sitzungsmanager ist nicht von den WebSphere-APIs abhängig und somit in der Lage, verschiedene Versionen von WebSphere Application Server und auch Anwendungsserverumgebungen anderer Anbieter zu unterstützen.

Der HTTP-Sitzungsmanager stellt Sitzungsverwaltungsfunktionen für eine zugeordnete Anwendung zur Verfügung. Der Sitzungsmanager erstellt HTTP-Sitzungen und verwaltet die Lebenszyklen von HTTP-Sitzungen, die der Anwendung zugeordnet sind. Zu diesen Verwaltungsaktivitäten für den Lebenszyklus gehören das Ungültigmachen von Sitzungen auf der Basis eines Zeitlimits oder eines expliziten

Servlet- oder JSP-Aufrufs (JavaServer Pages) und der Aufruf von Sitzungs-Listenern, die der Sitzung bzw. der Webanwendung zugeordnet sind. Der Sitzungsmanager definiert seine Sitzung in einer ObjectGrid-Instanz als persistent. Diese Instanz kann eine lokale speicherinterne Instanz oder eine vollständig replizierte und partitionierte Clusterinstanz sein. Die Verwendung der letzteren Topologie ermöglicht dem Sitzungsmanager, die Unterstützung für HTTP-Sitzungs-Failover bereitzustellen, wenn Anwendungsserver heruntergefahren oder unerwartet beendet werden. Der Sitzungsmanager kann auch in Umgebungen eingesetzt werden, die keine Affinität unterstützen, wenn die Affinität nicht durch eine Lastausgleichsfunktionsschicht erzwungen wird, die Anforderungen auf die Anwendungsserver verteilt.

Einsatzszenarios

Der Sitzungsmanager kann in den folgenden Szenarios verwendet werden:

- In Umgebungen, die Anwendungsserver verschiedener Versionen von WebSphere Application Server verwenden, z. B. in einem klassischen Migrationsszenario.
- In Implementierungen, die Anwendungsserver verschiedener Anbieter verwenden. Ein Beispiel hierfür sind Anwendungen, die unter Open-Source-Anwendungsservern entwickelt werden und dann in WebSphere Application Server eingesetzt werden. Ein weiteres Beispiel sind Anwendungen, die von der Bereitstellung auf die Produktion hochgestuft werden. Eine nahtlose Migration dieser Anwendungsserverversionen ist möglich, während alle HTTP-Sitzungen aktiv und bedient werden.
- In Umgebungen, die erfordern, dass der Benutzer Sitzungen mit höheren Servicequalitätsstufen und besseren Garantien für die Sitzungsverfügbarkeit beim Server-Failover als den Standardservicequalitätsstufen von WebSphere Application Server als persistent definieren.
- In einer Umgebung, in der die Sitzungsaffinität nicht garantiert werden kann, oder in Umgebungen, in denen die Affinität von einer anbieterspezifischen Lastausgleichsfunktion verwaltet wird und der Affinitätsmechanismus an diese Lastausgleichsfunktion angepasst werden muss.
- In einer Umgebung, in der die Sitzungsverwaltung und -speicherung in einen externen Java-Prozess ausgelagert werden muss.
- In mehreren Zellen, in denen das Sitzungs-Failover zwischen den Zellen aktiviert werden muss.

Funktionsweise des Sitzungsmanagers

Der Sitzungsmanager integriert sich in den Anforderungspfad in Form eines Servlet-Filters. Mit Hilfe von Tools, die mit WebSphere eXtreme Scale bereitgestellt werden, können Sie diesen Servlet-Filter jedem Webmodul hinzufügen. Sie können den Filter auch manuell dem Webimplementierungsdeskriptor Ihrer Anwendung hinzufügen. Dieser Filter empfängt die Anforderung vor dem Servlet bzw. den JSP-Dateien in der Zielanwendung. Der Filter fängt die HTTPRequest- und HTTPResponse-Objekte ab und erstellt ein Wrapper-Objekt in seiner Implementierung.

Diese Filterimplementierung fängt alle Aufrufe für die HTTPRequest- und HTTPResponse-Objekte ab, die sich auf die HTTP-Sitzung beziehen. Diese Aufrufe werden vom Sitzungsmanager bearbeitet und nicht an den Basissitzungsmanager in der zugrunde liegenden Java-EE-Serverimplementierung übergeben. Der Sitzungsmanager erstellt und verwaltet Sitzungen und ihre Lebenszyklen. Dazu gehört, dass er Sitzungen nach Ablauf eines festgelegten Zeitlimits ungültig macht und

Listener startet, wenn Sitzungen ungültig gemacht werden oder andere Lebenszyklusereignisse eintreten.

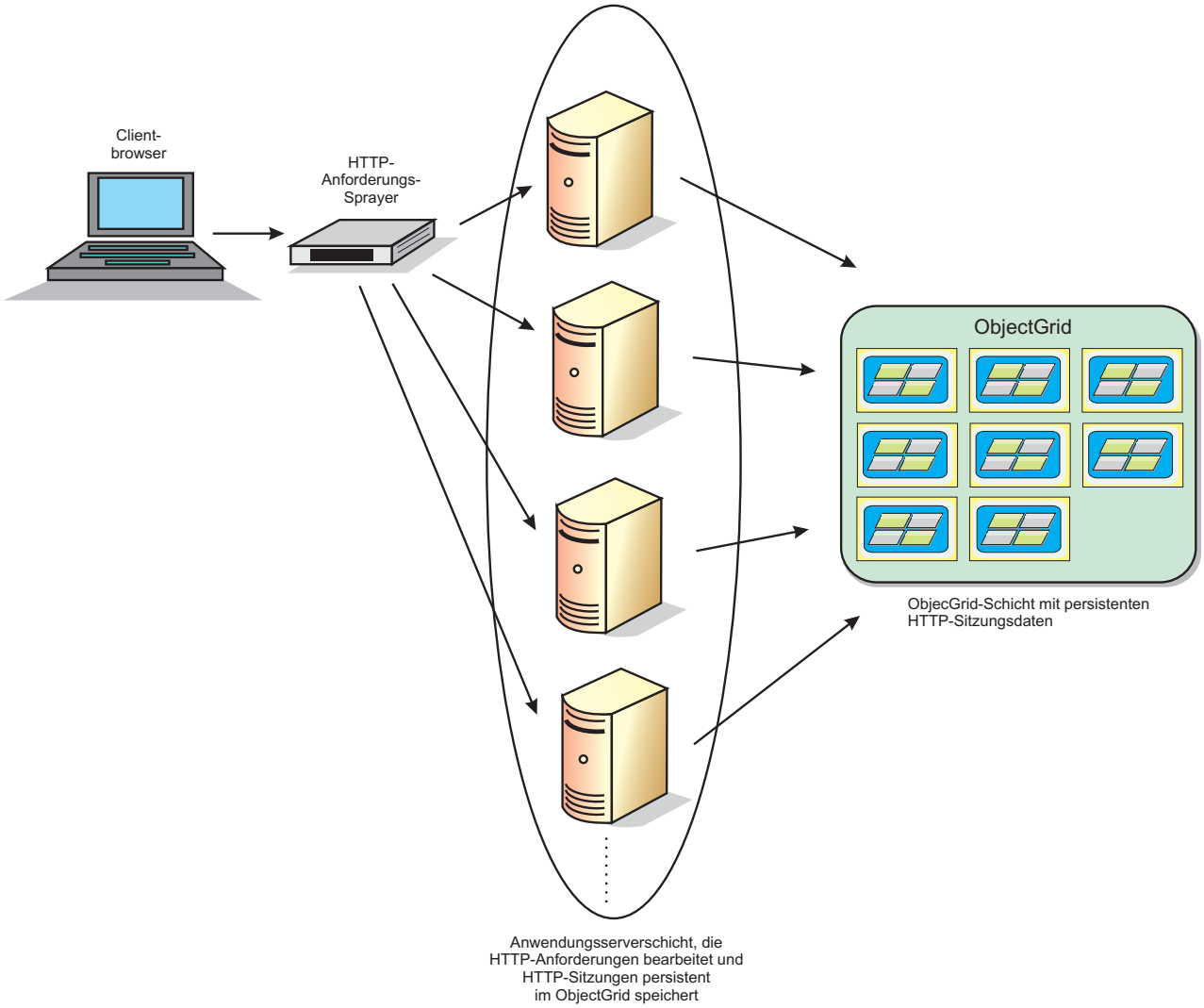


Abbildung 6. Topologie für die HTTP-Sitzungsverwaltung mit einer fernen Containerkonfiguration

Implementierungstopologien

Es gibt zwei dynamische Implementierungsszenarios für die Konfiguration des Sitzungsmanagers:

- **Integrierte eXtreme-Scale-Container mit Netzanschluss**

  In diesem Szenario werden die Server von eXtreme Scale in denselben Prozessen wie die Servlets zusammengefasst. Der Sitzungsmanager kann direkt mit der lokalen ObjectGrid-Instanz kommunizieren, wodurch teure Verzögerungen bei der Netzübertragung vermieden werden.
- **Ferne eXtreme-Scale-Container mit Netzanschluss**

  In diesem Szenario werden die Server von eXtreme Scale in Prozessen ausgeführt, die von dem Prozess abgesondert sind, in dem die Servlets ausgeführt werden. Der Sitzungsmanager kommuniziert mit einer fernen Instanz von eXtreme Scale.

eXtreme-Scale-Sitzung in einer Sitzungsmanageranwendung abrufen

```
public void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException {

        HttpSession session = req.getSession(true);

        System.out.println("calling getSession");
        // call getAttribute("com.ibm.websphere.objectgrid.session")
    // to get the ObjectGrid session instance
        Session ogSession = (Session)session.getAttribute
      ("com.ibm.websphere.objectgrid.session");

        System.out.println("ogSession = "+ogSession);
    }
```

Alle Änderungen, die über die vom Methodenaufruf ″getAttribute″ zurückgegebenen Sitzung vorgenommen werden, werden festgeschrieben, wenn die zugrunde liegende Sitzung festgeschrieben wird.

Zugehörige Tasks

„Sitzungsmanager von WebSphere eXtreme Scale für die Zusammenarbeit mit WebSphere Application Server konfigurieren”
WebSphere Application Server stellt zwar Funktionen für die Sitzungsverwaltung bereit, aber diese Unterstützung ist unter sehr hohen Anforderungslasten nicht skalierbar. Im Lieferumfang von WebSphere eXtreme Scale ist eine Implementierung für Sitzungsverwaltung enthalten, die den Standardsitzungsmanager für Webcontainer überschreibt und eine bessere Skalierbarkeit und leistungsfähigere Konfigurationsoptionen bietet.

„HTTP-Sitzungsmanager für die Zusammenarbeit mit WebSphere Application Server Community Edition konfigurieren” auf Seite 256
WebSphere Application Server Community Edition kann den Sitzungsstatus zwar freigeben, aber nicht auf effiziente und skalierbare Weise. WebSphere eXtreme Scale stellt eine verteilte Persistenzschicht mit hoher Leistung bereit, die zum Replizieren des Status verwendet werden kann, sich aber nicht problemlos mit Anwendungsservern außerhalb von WebSphere Application Server integrieren lässt. Sie können diese beiden Produkte integrieren, um eine skalierbare Lösung für die Sitzungsverwaltung zu erhalten. Sie können die modulare Infrastruktur von WebSphere Application Server Community Edition, GBeans, verwenden, um WebSphere eXtreme Scale als Persistenzmechanismus für den Sitzungsstatus zu integrieren.

Zugehörige Verweise

„Initialisierungsparameter für den Servlet-Kontext” auf Seite 253
Die folgende Liste mit Initialisierungsparametern für den Servlet-Kontext kann in der Eigenschaftendatei in den script- oder Ant-basierten Splicing-Methoden angegeben werden.

Sitzungsmanager von WebSphere eXtreme Scale für die Zusammenarbeit mit WebSphere Application Server konfigurieren

WebSphere Application Server stellt zwar Funktionen für die Sitzungsverwaltung bereit, aber diese Unterstützung ist unter sehr hohen Anforderungslasten nicht skalierbar. Im Lieferumfang von WebSphere eXtreme Scale ist eine Implementierung für Sitzungsverwaltung enthalten, die den Standardsitzungsmanager für Webcontainer überschreibt und eine bessere Skalierbarkeit und leistungsfähigere Konfigurationsoptionen bietet.

Warum und wann dieser Vorgang ausgeführt wird

Der HTTP-Sitzungsmanager von WebSphere eXtreme Scale unterstützt integrierte und ferne Server für das Caching.

Szenario mit integriertem Sitzungsmanager

Im Szenario mit dem integrierten Sitzungsmanager werden die WebSphere eXtreme Scale-Server in denselben Prozessen wie die Servlets ausgeführt. Der Sitzungsmanager kann direkt mit der lokalen ObjectGrid-Instanz kommunizieren, wodurch teure Verzögerungen bei der Netzübertragung vermieden werden.

Wenn Sie WebSphere Application Server verwenden, kopieren Sie die mitgelieferten Dateien `ObjectGrid-Stammverzeichnis/session/samples/objectGrid.xml` und `ObjectGrid-Stammverzeichnis/session/samples/objectGridDeployment.xml` in die META-INF-Verzeichnisse Ihrer WAR-Dateien. eXtreme Scale erkennt diese Dateien beim Anwendungsstart automatisch und startet automatisch die eXtreme-Scale-Container in demselben Prozess wie den Sitzungsmanager.

Sie können die Datei `objectGridDeployment.xml` ändern, abhängig davon, ob Sie synchrone oder asynchrone Replikation verwenden möchten und wie viele Replikate konfiguriert werden sollen.

Szenario mit fernen Servern

Im Szenario mit fernen Servern werden die eXtreme-Scale-Server in anderen Prozessen als die Servlets ausgeführt. Der Sitzungsmanager kommuniziert mit einem fernen eXtreme-Scale-Server. Wenn Sie einen fernen, über ein Netz verbundenen eXtreme-Scale-Server verwenden möchten, muss der Sitzungsmanager mit Hostnamen und Ports des eXtreme-Scale-Server-Katalogserviceclusters konfiguriert werden. Der Sitzungsmanager verwendet dann eine eXtreme-Scale-Clientverbindung, um mit dem Katalogserver und den eXtreme-Scale-Servern zu kommunizieren.

Wenn die eXtreme-Scale-Server in unabhängigen, eigenständigen Prozessen gestartet werden, starten Sie die eXtreme-Scale-Container mit den Dateien `objectGridStandAlone.xml` und `objectGridDeploymentStandAlone.xml`, die im Verzeichnis "samples" des Sitzungsmanagers bereitgestellt werden.

1. **Verwendung von WebSphere Application Server oder WebSphere Application Server Network Deployment ohne WebSphere eXtreme Scale-Installation:** Installieren Sie die Bibliotheken für eXtreme Scale.

   Kopieren Sie die Dateien `ObjectGrid-Stammverzeichnis/session/lib/sessionobjectgrid.jar` und `ObjectGrid-Stammverzeichnis/lib/wsobjectgrid.jar` in den Klassenpfad der Anwendungsserver, in denen die Anwendung mit der neuen HTTP-Sitzungsverwaltung ausgeführt wird. Speichern Sie diese JAR-Dateien (Java-Archiv) im Ordner `<WAS_HOME>/lib`, und starten Sie anschließend den Server erneut, damit diese Klassen sichtbar werden.
2. Fügen Sie Ihre Anwendung so zusammen, dass sie den Sitzungsmanager verwenden kann. Wenn Sie den Sitzungsmanager verwenden möchten, müssen Sie die entsprechenden Filterdeklarationen den Webimplementierungsdeskriptoren für die Anwendung hinzufügen. Außerdem werden die Konfigurationsparameter des Sitzungsmanagers in Form von Initialisierungsparametern für den Servlet-Kontext in den Implementierungsdeskriptoren an den Sitzungsmanager übergeben. Es gibt drei Methoden, mit denen Sie diese Informationen in Ihre Anwendung einführen können:
   - **Script addObjectGridFilter**

     Verwenden Sie ein Befehlszeilenscript, das mit eXtreme Scale bereitgestellt wird, um eine Anwendung mit Filterdeklarationen und Konfigurationen in Form von Initialisierungsparametern für den Servlet-Kontext zusammenzufügen. Dieses Script, das den Namen `ObjectGrid-Stammverzeichnis/bin/addObjectGridFilter.sh` bzw. `ObjectGrid-Stammverzeichnis/bin/`

`addObjectGridFilter.bat` hat, akzeptiert zwei Parameter: die Anwendung (absoluter Pfad der EAR-Datei), die zusammengefügt werden muss, und den absoluten Pfad der Eigenschaftendatei, die die verschiedenen Konfigurationsparameter enthält. Das Syntaxformat dieses Scripts ist wie folgt:

```
addObjectGridFilter.bat [Position der EAR-Datei] [Position der Eigenschaftendatei]
```

```
addObjectGridFilter.sh [Position der EAR-Datei] [Position der Eigenschaftendatei]
```

Beispiel mit einer Installation von eXtreme Scale in WebSphere Application Server unter UNIX:

a. `cd ObjectGrid-Stammverzeichnis/optionalLibraries/ObjectGrid/session/bin`
b. `addObjectGridFilter.sh /tmp/mySessionTest.ear WAS-Stammverzeichnis/optionalLibraries/ObjectGrid/session/samples/splicer.properties`

Beispiel mit einer Installation von eXtreme Scale in einem eigenständigen Verzeichnis unter UNIX:

a. `cd ObjectGrid-Stammverzeichnis/session/bin`
b. `addObjectGridFilter.sh /tmp/mySessionTest.ear ObjectGrid-Stammverzeichnis/session/samples/splicer.properties`

Der Servlet-Filter, der eingefügt wird, verwaltet die Standardwerte für Konfigurationswerte. Sie können diese Standardwerte mit Konfigurationsoptionen überschreiben, die Sie in der Eigenschaftendatei im zweiten Argument angeben. Eine Liste der Parameter, die Sie verwenden können, finden Sie im Abschnitt „Initialisierungsparameter für den Servlet-Kontext" auf Seite 253.

Sie können die Musterdatei `splicer.properties` ändern und verwenden, die mit der Installation von eXtreme Scale bereitgestellt wird. Sie können auch das Script "addObjectGridServlets" verwenden, das den Sitzungsmanager einfügt, indem Sie jedes Servlet erweitern. Das empfohlene Script ist jedoch "addObjectGridFilter".

- **Ant-Build-Script**

  Im Lieferumfang von WebSphere eXtreme Scale ist eine Datei `build.xml` enthalten, das von Apache Ant verwendet werden kann, und im Ordner `WAS-Stammverzeichnis/bin` einer Installation von WebSphere Application Server enthalten ist. Die Datei `build.xml` kann bearbeitet werden, um die Konfigurationseigenschaften des Sitzungsmanagers zu ändern. Die Konfigurationseigenschaften sind identisch mit den Eigenschaftsnamen in der Datei `splicer.properties`. Nachdem Sie die Datei `build.xml` geändert haben, wird der Ant-Prozess mit ant.sh, ws_ant.sh (UNIX) bzw. ant.bat, ws_ant.bat (Windows) aufgerufen.

- **Webdeskriptor manuell aktualisieren**

  Editieren Sie die mit der Webanwendung gepackte Datei `web.xml`, um die Filterdeklaration, die Servlet-Zuordnung und die Initialisierungsparameter für den Servlet-Kontext zu integrieren. Sie sollten diese Methode nicht verwenden, weil sie fehleranfällig ist.

Eine Liste der Parameter, die Sie verwenden können, finden Sie im Abschnitt „Initialisierungsparameter für den Servlet-Kontext" auf Seite 253.

3. Implementieren Sie die Anwendung. Implementieren Sie die Anwendung. Führen Sie dazu die Schritte aus, die Sie gewöhnlich für einen Server oder Cluster verwenden. Nach der Implementierung der Anwendung können Sie die Anwendung starten.

4. Greifen Sie auf die Anwendung zu. Sie können jetzt auf die Anwendung zugreifen, die mit dem Sitzungsmanager und mit WebSphere eXtreme Scale interagiert.

Nächste Maßnahme

Sie können die meisten Konfigurationsparameter des Sitzungsmanagers ändern, wenn Sie Ihre Anwendung für die Verwendung des Sitzungsmanagers instrumentieren. Zu diesen Attributen gehören Varianten des Replikationstyps (synchron oder asynchron), die Länge der Sitzungs-ID, die Größe der speicherinternen Sitzungstabelle usw. Abgesehen von den Attributen, die während der Instrumentierung der Anwendung geändert werden können, sind die einzigen Attribute, die Sie nach der Anwendungsimplementierung ändern können, die Attribute, die sich auf die WebSphere eXtreme Scale-Serverclustertopologie beziehen, und die Art und Weise, in der Clients (Sitzungsmanager) eine Verbindung zu diesen Servern herstellen.

Zugehörige Konzepte

„Verwaltung von HTTP-Sitzungen" auf Seite 247
In einer Umgebung mit WebSphere Application Server Version 5 oder höher kann der mit WebSphere eXtreme Scale bereitgestellte HTTP-Sitzungsmanager den Standardsitzungsmanager im Anwendungsserver außer Kraft setzen.

Zugehörige Verweise

„Initialisierungsparameter für den Servlet-Kontext"
Die folgende Liste mit Initialisierungsparametern für den Servlet-Kontext kann in der Eigenschaftendatei in den script- oder Ant-basierten Splicing-Methoden angegeben werden.

Initialisierungsparameter für den Servlet-Kontext

Die folgende Liste mit Initialisierungsparametern für den Servlet-Kontext kann in der Eigenschaftendatei in den script- oder Ant-basierten Splicing-Methoden angegeben werden.

Parameter

affinityManager
Gibt das vollständig qualifizierte Paket und den Namen der Java-Klasse eines Servlet-Filter-Plug-ins für die Vereinfachung der Routing-Affinität von HTTP-Sitzungen an. Dieser Wert wird für WebSphere Application Server ignoriert, weil die Affinitätsunterstützung integriert ist. Geben Sie einen der folgenden Werte an:

- **Keine Affinität (Standardeinstellung):** com.ibm.ws.httpsession.NoAffinityManager
- **Anbieterspezifischer Affinitätsmechanismus:** com.ibm.ws.httpsession.AssumeAffinityManager
- **Neuen Affinitätsmanager erstellen:** Verwenden Sie die Schnittstelle "com.ibm.wsspi.session.ISessionAffinityManager".

persistenceMechanism
Gibt einen Zeichenfolgewert an, der definiert, wie die Sitzung in eXtreme Scale gespeichert wird. Geben Sie einen der folgenden Werte an:

- **ObjectGridStore**: Jedes Sitzungsattribut wird als eigener Eintrag in der eXtreme-Scale-Tabelle gespeichert.
- **ObjectGridAtomicSessionStore**: Die gesamte Sitzung wird als ein einziger Eintrag in der eXtreme-Scale-Tabelle gespeichert.

objectGridName
Gibt einen Zeichenfolgewert an, der den Namen der Tabelle definiert, die für eine bestimmte Webanwendung verwendet wird. Der Standardname ist der Webanwendungsname, der aus dem ServletContext-Wert abgeleitet wird. Wenn Sie diesen Parameter setzen, wird dieser Wert überschrieben.

catalogHostPort
Gibt Verbindungsinformationen für den Katalogserver an. Der Wert muss im Format `Host:Port<,Host:Port>` angegeben werden. Diese Liste kann beliebig lang sein, und es wird die erste funktionsfähige Adresse verwendet. Diese Eigenschaft wird nur für das ferne eXtreme-Scale-Szenario mit Netzanschluss benötigt.

replicationType
Ein Zeichenfolgewert, der definiert, wie Aktualisierungen in Sitzungen in eXtreme Scale geschrieben werden. Die gültigen Werte sind `asynchronous` und `synchronous`. Der Standardwert ist `asynchronous`, der nur anwendbar ist, wenn Affinität verwendet wird. Andernfalls wird nur ″synchronous″ unterstützt.

replicationInterval
Ein ganzzahliger Wert, der das Intervall (in Sekunden) angibt, in dem aktualisierte Sitzungen in eXtreme Scale geschrieben werden, wenn der Parameter ″replicationType″ auf `asynchronous` gesetzt ist. Der Standardwert sind 10 Sekunden.

sessionTableSize
Ein ganzzahliger Wert, der die Anzahl der Sitzungen definiert, die in eXtreme Scale gespeichert und zusätzlich über den Servlet-Filter im Speicher gehalten werden. Der Standardwert ist 1000.

defaultSessionTimeout
Ein ganzzahliger Wert, der die zulässige Inaktivitätszeit (z. B. keine Zugriffe über ein Servlet) einer Sitzung definiert, bevor die Sitzung ungültig gemacht und aus dem System entfernt wird. Der Standardwert sind 30 Minuten.

sessionIDLength
Ein ganzzahliger Wert, der die Länge der Zeichenfolgekennungen definiert, die für HTTP-Sitzungen erstellt werden. Der Standardwert ist 23.

shareSessionsAcrossWebApps
Gibt einen Zeichenfolgewert true oder false an. Der Standardwert ist ″false″. Gemäß Servlet-Spezifikation können HTTP-Sitzungen nicht von mehreren Webanwendungen gemeinsam genutzt werden. Für die Unterstützung der gemeinsamen Nutzung wird eine Erweiterung der Servlet-Spezifikation bereitgestellt.

cookieName
Gibt einen Zeichenfolgewert an, der den Namen des Cookies für diese Webanwendung definiert. Der Standardwert ist JSESSIONID. Wenn Sie einen eindeutigen Cookie-Namen verwenden möchten, fügen Sie diese Eigenschaft der Datei `splicer.properties` hinzu.

Zugehörige Konzepte

„Verwaltung von HTTP-Sitzungen" auf Seite 247
In einer Umgebung mit WebSphere Application Server Version 5 oder höher kann der mit WebSphere eXtreme Scale bereitgestellte HTTP-Sitzungsmanager den Standardsitzungsmanager im Anwendungsserver außer Kraft setzen.

Zugehörige Tasks

„Sitzungsmanager von WebSphere eXtreme Scale für die Zusammenarbeit mit WebSphere Application Server konfigurieren" auf Seite 250
WebSphere Application Server stellt zwar Funktionen für die Sitzungsverwaltung bereit, aber diese Unterstützung ist unter sehr hohen Anforderungslasten nicht skalierbar. Im Lieferumfang von WebSphere eXtreme Scale ist eine Implementierung für Sitzungsverwaltung enthalten, die den Standardsitzungsmanager für Webcontainer überschreibt und eine bessere Skalierbarkeit und leistungsfähigere Konfigurationsoptionen bietet.

„HTTP-Sitzungsmanager für die Zusammenarbeit mit WebSphere Application Server Community Edition konfigurieren" auf Seite 256
WebSphere Application Server Community Edition kann den Sitzungsstatus zwar freigeben, aber nicht auf effiziente und skalierbare Weise. WebSphere eXtreme Scale stellt eine verteilte Persistenzschicht mit hoher Leistung bereit, die zum Replizieren des Status verwendet werden kann, sich aber nicht problemlos mit Anwendungsservern außerhalb von WebSphere Application Server integrieren lässt. Sie können diese beiden Produkte integrieren, um eine skalierbare Lösung für die Sitzungsverwaltung zu erhalten. Sie können die modulare Infrastruktur von WebSphere Application Server Community Edition, GBeans, verwenden, um WebSphere eXtreme Scale als Persistenzmechanismus für den Sitzungsstatus zu integrieren.

WebSphere eXtreme Scale für die Verwaltung von SIP-Sitzungen verwenden

Sie können WebSphere eXtreme Scale als Alternative zum Datenreplikationsservice (DRS) als SIP-Replikationsmechanismus (Session Initiation Protocol) für die Replikation von SIP-Sitzungen verwenden.

Konfiguration der SIP-Sitzungsverwaltung

Wenn Sie WebSphere eXtreme Scale als SIP-Replikationsmechanismus verwenden möchten, definieren Sie die angepasste Eigenschaft "com.ibm.sip.ha.replicator.type". Wählen Sie in der Administrationskonsole für jeden Server, dem Sie die angepasste Eigenschaft hinzufügen möchten, **Anwendungsserver** → ***mein_Anwendungsserver*** → **SIP-Container** → **Angepasste Eigenschaften** aus. Geben Sie `com.ibm.sip.ha.replicator.type` im Feld "Name" und `OBJECTGRID` im Feld "Wert" ein.

Verwenden Sie die folgenden Eigenschaften, um das Verhalten des ObjectGrids anzupassen, das zum Speichern von SIP-Sitzungen verwendet wird. Klicken Sie in der Administrationskonsole für jeden Server, dem Sie die angepasste Eigenschaft hinzufügen möchten, auf **Anwendungsserver** → ***mein_Anwendungsserver*** → **SIP-Container** → **Angepasste Eigenschaften**. Füllen Sie die Felder **Name** und **Wert** aus. Für jeden Server müssen dieselben Eigenschaften definiert werden, damit sie ordnungsgemäß funktionieren.

Tabelle 8. Angepasste Eigenschaften für die SIP-Sitzungsverwaltung mit ObjectGrid

Eigenschaft	Wert	Standardwert
com.ibm.sip.ha.replicator.type	`OBJECTGRID`: ObjectGrid als SIP-Sitzungsspeicher verwenden	

Tabelle 8. Angepasste Eigenschaften für die SIP-Sitzungsverwaltung mit ObjectGrid (Forts.)

Eigenschaft	Wert	Standardwert
min.synchronous.replicas	Mindestanzahl synchroner Replikate	0
max.synchronous.replicas	Maximale Anzahl synchroner Replikate	0
max.asynchronous.replicas	Maximale Anzahl asynchroner Replikate	1
auto.replace.lost.shards	Weitere Informationen finden Sie im Abschnitt „Konfiguration der Implementierungstopologie" auf Seite 149.	true
development.mode	• `true`: Verwendung aktiver Replikate auf demselben Knoten wie primäre Entität zulassen. • `false`: Replikate dürfen sich nicht zusammen mit der primären Entität auf demselben Knoten befinden.	false

HTTP-Sitzungsmanager für die Zusammenarbeit mit WebSphere Application Server Community Edition konfigurieren

WebSphere Application Server Community Edition kann den Sitzungsstatus zwar freigeben, aber nicht auf effiziente und skalierbare Weise. WebSphere eXtreme Scale stellt eine verteilte Persistenzschicht mit hoher Leistung bereit, die zum Replizieren des Status verwendet werden kann, sich aber nicht problemlos mit Anwendungsservern außerhalb von WebSphere Application Server integrieren lässt. Sie können diese beiden Produkte integrieren, um eine skalierbare Lösung für die Sitzungsverwaltung zu erhalten. Sie können die modulare Infrastruktur von WebSphere Application Server Community Edition, GBeans, verwenden, um WebSphere eXtreme Scale als Persistenzmechanismus für den Sitzungsstatus zu integrieren.

Vorbereitungen

Sie müssen Geronimo oder WebSphere Application Server Community Edition und WebSphere eXtreme Scale auf Ihrem System dekomprimieren und installieren.

Warum und wann dieser Vorgang ausgeführt wird

Das Backbone von WebSphere Application Server Community Edition, der Kernel, stützt sich auf Module, so genannte GBeans, die Funktionen und Leistungsmerkmale bereitstellen.

1. Verteilen Sie die GBeans.
   a. Navigieren Sie zum Verzeichnis *eXtreme-Scale-Stammverzeichnis*`/wasce/bin`.
   b. Führen Sie das Script "addToCeRepository" für das System aus, das Sie verwenden.
      - addToRepository.sh ce-Stammverzeichnis
      - addToRepository.bat ce-Stammverzeichnis
2. Editieren Sie die Konfigurationsdatei. Konfigurieren Sie nach der Verteilung der GBeans WebSphere Application Server Community Edition so, dass das Produkt weiß, dass sie beim Serverstart gestartet werden sollen. Sie können die GBeans registrieren und mit konfigurierbaren Argumenten angeben, die zur Laufzeit mit der Datei `config.xml` verwendet werden können.
   a. Navigieren Sie zum Verzeichnis `WasCe-Stammverzeichnis/var/config`.
   b. Öffnen Sie die Datei `config.xml` in einem Texteditor.
   c. Suchen Sie die folgende Zeile in der Datei `config.xml`:

```
<module name="org.apache.geronimo.plugins/mconsole-ds/2.1.1/car"/>
 </attributes>
```

d. Fügen Sie den folgenden Auszug vor dem Tag `</attributes>` ein.

```
<module name="com.ibm.websphere.objectgrid/gbean/1.0/car">
   <gbean name="objectgrid/BringupPlugin">
   <attribute name="objectgridHome">c:\objectgrid</attribute>
   <attribute name="geronimoHome">C:\websphereCE</attribute>
   <attribute name="serverName">server1</attribute>
   <attribute name="catalogServiceEndPoints">host:port</attribute>
   <attribute name="replicationDisabled">false</attribute>
   <attribute name="traceSpecification">*=all=disabled</attribute>
   </gbean>
</module>
```

Die Werte für die GBean-Parameter im vorherigen Auszug sind Beispiele. Sie können diese Werte konfigurieren.

objectgridHome
Gibt das Installationsverzeichnis des ObjectGrids an.

geronimoHome -
Gibt das Installationsverzeichnis von WebSphere Application Server Community Edition bzw. Apache Geronimo an.

serverName
Gibt einen Servernamen an, der für jede Serverinstanz von WebSphere Application Server Community Edition und eXtreme Scale eindeutig sein muss.

catalogServiceEndPoints
Gibt den Host und den Port des zu verwendenden Katalogservers an.

replicationDisabled
Gibt an, ob die Replikation des Sitzungsstatus aktiviert wird.

traceSpecification
Gibt eine Trace-Zeichenfolge für den eXtreme-Scale-Container in der Java Virtual Machine (JVM) an.

3. Starten Sie einen Katalogserver. Ihr Image von WebSphere Application Server Community Edition kann einen eXtreme-Scale-Server beim Serverstart erfolgreich in seiner JVM starten. Für das Bootstrapping von eXtreme Scale müssen Sie jedoch zuerst einen Katalogserver starten. Der Katalogserver entspricht einem Deployment Manager in einer Topologie von WebSphere Application Server und muss vor WebSphere Application Server Community Edition gestartet werden. Nach dem Starten des Katalogservers geben Sie den Hostnamen und den Port für den Katalogserver in der Datei `config.xml` im GBean-Eintrag an. Weitere Informationen hierzu finden Sie im Abschnitt „Katalogservice in einer eigenständigen Umgebung starten" auf Seite 229. Lesen Sie unbedingt den Abschnitt zum Binden des ORB-Ports an eine bestimmte Hostname/Port-Kombination. Sie müssen diese Endpunkte in der GBean-Konfiguration angeben. Wenn beim Serverstart ein Portkonflikt auftritt, müssen Sie die Datei `ce-Stammverzeichnis/var/config/config-substitutions.properties` öffnen und den Wert des Schlüssels "NamingPort" in einen anderen Wert als `1099` ändern.
4. Starten Sie WebSphere Application Server Community Edition, um das Bootstrapping zu testen.
   a. Navigieren Sie zum Verzeichnis `WasCe-Stammverzeichnis/bin`.
   b. Setzen Sie die Umgebungsvariable JAVA_HOME.

- `export JAVA_HOME=javaHome`
- `set JAVA_HOME=javaHome`

c. Führen Sie den Startbefehl aus.

- `geronimo.sh run`
- `geronimo.bat run`

Der Server sollte jetzt gestartet werden und mit der Initialisierung seiner Komponenten bzw. GBeans beginnen. Das eXtreme-Scale-Plug-in ist das letzte Plug-in, das geladen wird, und gibt den erforderlichen Start-Trace und Informationsanweisungen aus.

Zugehörige Konzepte

„Verwaltung von HTTP-Sitzungen" auf Seite 247
In einer Umgebung mit WebSphere Application Server Version 5 oder höher kann der mit WebSphere eXtreme Scale bereitgestellte HTTP-Sitzungsmanager den Standardsitzungsmanager im Anwendungsserver außer Kraft setzen.

Zugehörige Verweise

„Initialisierungsparameter für den Servlet-Kontext" auf Seite 253
Die folgende Liste mit Initialisierungsparametern für den Servlet-Kontext kann in der Eigenschaftendatei in den script- oder Ant-basierten Splicing-Methoden angegeben werden.

WebSphere eXtreme Scale für die Replikation des Sitzungsstatus in WebSphere Application Server Community Edition verwenden

Jede Webanwendung, die in WebSphere Application Server Community Edition implementiert wird, muss mit der Spezifikation Java Platform, Enterprise Edition und der Deskriptordatei `geronimo-web.xml` kompatibel sein. Diese Datei enthält Zugriffs- und Abhängigkeitsinformationen, die für die Laufzeitumgebung von WebSphere Application Server Community Edition spezifisch sind. Damit eine Java-EE-Webanwendung das Feature für Sitzungsreplikation von WebSphere eXtreme Scale verwenden kann, muss die Webawendung Daten zum eXtreme-Scale-Plug-in und sitzungsspezifische Attribute übergeben.

Vorbereitungen

Dieser Prozess ist so einfach, dass er ohne IDE oder Dienstprogramme auf jedem Betriebssystem ausgeführt werden kann.

Warum und wann dieser Vorgang ausgeführt wird

Sie können eine Basiswebanwendung konfigurieren, die später die Sitzungspersistenz in eXtreme Scale nutzt, die mitrechnet, wie oft ein Benutzer auf ein bestimmtes Servlet zugreift und diese Informationen speichert.

1. Erstellen Sie die Musterwebanwendung.
   a. Erstellen Sie ein Verzeichnis an einer beliebigen Position in Ihrem Dateisystem. Sie können beispielsweise ein Verzeichnis mit dem Namen *app_home* erstellen.
   b. Navigieren Sie zum Verzeichnis *app_home*, und erstellen Sie die folgende Verzeichnisstruktur:

```
> app_home
--> META-INF
--> WEB-INF
-----> lib
-----> classes
```

c. Erstellen Sie über das Verzeichnis *app_home*/*WEB-INF* eine Datei `web.xml`. Kopieren und fügen Sie den folgenden Code in die Datei `web.xml` ein. Stellen Sie sicher, dass der Datei `web.xml` alle Kontextparameter hinzugefügt werden, oder der Filter funktioniert nicht ordnungsgemäß.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="sample" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
  http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
 <display-name>
 sample</display-name>
 <context-param>
    <param-name>shareSessionsAcrossWebApps</param-name>
    <param-value>false</param-value>
  </context-param>
  <context-param>
      <param-name>sessionIDLength</param-name>
      <param-value>23</param-value>
  </context-param>
  <context-param>
      <param-name>sessionTableSize</param-name>
      <param-value>1000</param-value>
  </context-param>
  <context-param>
      <param-name>replicationInterval</param-name>
      <param-value>10</param-value>
  </context-param>
  <context-param>
      <param-name>replicationType</param-name>
      <param-value>synchronous</param-value>
  </context-param>
  <context-param>
      <param-name>defaultSessionTimeout</param-name>
      <param-value>30</param-value>
  </context-param>
  <context-param>
      <param-name>affinityManager</param-name>
      <param-value>com.ibm.ws.httpsession.NoAffinityManager</param-value>
  </context-param>
  <context-param>
      <param-name>useURLEncoding</param-name>
      <param-value>true</param-value>
  </context-param>
  <context-param>
      <param-name>objectGridName</param-name>
      <param-value>session</param-value>
  </context-param>
  <context-param>
      <param-name>persistenceMechanism</param-name>
      <param-value>ObjectGridStore</param-value>
  </context-param>
  <filter>
      <filter-name>HttpSessionFilter</filter-name>
      <filter-class>com.ibm.ws.httpsession.HttpSessionFilter</filter-class>
  </filter>
  <filter-mapping>
      <filter-name>HttpSessionFilter</filter-name>
       <url-pattern>/*</url-pattern>
  </filter-mapping>
 <servlet>
  <description>
  </description>
  <display-name>
  SampleServlet</display-name>
  <servlet-name>SampleServlet</servlet-name>
  <servlet-class>
  SampleServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>SampleServlet</servlet-name>
  <url-pattern>/SampleServlet</url-pattern>
 </servlet-mapping>
 <welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>default.html</welcome-file>
```

```
  <welcome-file>default.htm</welcome-file>
  <welcome-file>default.jsp</welcome-file>
 </welcome-file-list>
</web-app>
```

d. Erstellen Sie über das Verzeichnis *app_home/WEB-INF* eine Datei `geronimo-web.xml`. Kopieren und fügen Sie den folgenden Code in die Datei `geronimo-web.xml` ein:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.1"
 xmlns:nam="http://geronimo.apache.org/xml/ns/naming-1.1"
 xmlns:sec="http://geronimo.apache.org/xml/ns/security-1.1"
 xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.1">
 <sys:environment>
  <sys:moduleId>
   <sys:groupId>com.ibm.websphere.objectgrid</sys:groupId>
   <sys:artifactId>sample</sys:artifactId>
   <sys:version>1.0</sys:version>
   <sys:type>war</sys:type>
  </sys:moduleId>
  <sys:dependencies>
   <sys:dependency>
    <sys:groupId>com.ibm.websphere.objectgrid
    </sys:groupId>
    <sys:artifactId>session</sys:artifactId>
    <sys:version>1.0</sys:version>
    <sys:type>jar</sys:type>
   </sys:dependency>
  </sys:dependencies>
 </sys:environment>
 <context-root>/sample</context-root>
</web-app>
```

e. Kopieren Sie die folgende Datei `SampleServlet.class` in das Verzeichnis *WEB-INF/classes*.

f. Navigieren Sie zum Verzeichnis *app_home*.

g. Führen Sie den folgenden Befehl aus, um Ihre WAR-Datei (Webarchiv) zu packen:

```
jar -cf sample.war
```

h. Ihre Webanwendung kann jetzt implementiert werden.

2. Implementieren Sie Ihre Webanwendung.

a. Melden Sie sich auf der Homepage von WebSphere Application Server Community Edition unter der angegebenen `Hostname:Port`-Kombination an. Standardmäßig kann die Homepage mit `http://localhost:8080/` aufgerufen werden.

b. Klicken Sie auf **Administrationskonsole**.

c. Geben Sie Ihren Benutzernamen und Ihr Kennwort ein, und klicken Sie auf **OK**. Standardmäßig werden die folgenden Berechtigungsnachweise verwendet:

- Benutzername: System
- Kennwort: Manager

d. Wählen Sie im Menü auf der linken Seite **Anwendungen → Neue Anwendung implementieren** aus.

e. Klicken Sie rechts neben dem Eintrag "Archiv" auf **Durchsuchen**, und wählen Sie die Datei `sample.war` aus, die Sie im vorherigen Abschnitt erstellt haben.

f. Lassen Sie das Feld **Plan** leer, weil Sie den Plan mit der Webanwendung gepackt haben.

g. Wählen Sie **Anwendung nach Installation starten** aus, und klicken Sie auf **Installieren**.

Nach der Installation der Anwendung wird eine Nachricht angezeigt, in der Ihnen mitgeteilt wird, dass die Installation erfolgreich war.

3. Testen Sie die Anwendung.

a. Navigieren Sie zum Kontextstammverzeichnis `/sample/SampleServlet`, um die Anwendung zu testen. Standardmäßig ist das Kontextstammverzeichnis `http://localhost:8080/sample/SampleServlet`.
b. Es sollte angezeigt werden, wie oft Sie dieses Servlet aufgerufen haben, gefolgt von der HttpSession-Implementierung. Der Eintrag sollte ″HttpSessionImpl″ lauten, wenn die Persistenzschicht von eXtreme Scale verwendet wird, bzw. ″StandardSessionFacade″, wenn der Standardmechanismus des Webcontainers zum Speichern von Sitzungen verwendet wird.

Kapitel 8. Implementierungsumgebung überwachen

Sie können Anwendungsprogrammierschnittstellen (API, Application Programming Interface), MBeans, Protokolle und Dienstprogramme verwenden, um die Leistung Ihrer Anwendungsumgebung zu überwachen.

Zugehörige Konzepte

„Übersicht über Statistiken"
Statistiken in WebSphere eXtreme Scale werden über eine interne Statistikstruktur erstellt. Die StatsAccessor-API, die PMI-Module (Performance Monitoring Infrastructure) und die MBean-API werden aus der internen Struktur erstellt.

„Tool eines anderen Anbieters" auf Seite 290
WebSphere eXtreme Scale kann mit verschiedenen gängigen Lösungen für die Unternehmensüberwachung überwacht werden. Es werden Plug-in-Agenten für IBM Tivoli Monitoring and Hyperic HQ bereitgestellt, die WebSphere eXtreme Scale mit Hilfe öffentlich zugänglicher Management-Beans überwachen. CA Wily Introscope verwendet die Java-Methodeninstrumentierung, um Statistiken zu erfassen.

Zugehörige Verweise

„Managed Beans (MBeans) für die Verwaltung der Umgebung verwenden" auf Seite 285
Sie können verschiedene Typen von JMX-Beans (Java Management Extensions) verwenden, um Implementierungen zu verwalten und zu überwachen. Jede MBean bezieht sich auf eine bestimmte Entität, z. B. eine Map, eXtreme Scale, einen Server, eine Replikationsgruppe oder ein Replikationsgruppen-Member.

Übersicht über Statistiken

Statistiken in WebSphere eXtreme Scale werden über eine interne Statistikstruktur erstellt. Die StatsAccessor-API, die PMI-Module (Performance Monitoring Infrastructure) und die MBean-API werden aus der internen Struktur erstellt.

Die folgende Abbildung zeigt eine allgemeine Konfiguration von Statistiken für eXtreme Scale.

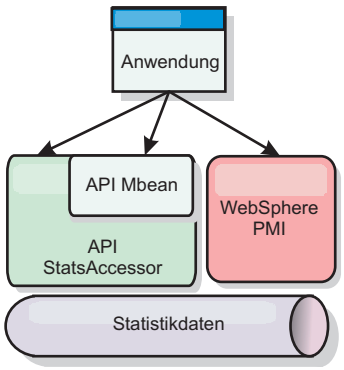


Abbildung 7. Übersicht über Statistiken

Jede dieser APIs bietet eine Sicht auf die Statistikstruktur, wird aber aus jeweils anderen Gründen verwendet:

- **Statistics-API**: Verwenden Sie die Statistics-API als generischen Suchmechanismus für Daten in der internen Statistikstruktur. Sie können die Statistics-API für Clients und integrierte Anwendungen verwenden. Die Statistics-API kann nicht verwendet werden, wenn Sie eine verteilte eXtreme-Scale-Umgebung haben.
- **MBean-API**: Die MBean-API ist ein spezifikationsbasierter Mechanismus für die Überwachung. Die MBean-API verwendet die Statistics-API und wird lokal in der JVM des Servers ausgeführt. Die API- und MBean-Strukturen sind so konzipiert, dass sie problemlos in die Dienstprogramme anderer Anbieter integriert werden können. Verwenden Sie die MBean-API, wenn Sie in einer verteilten eXtreme-Scale-Umgebung arbeiten.
- **PMI-Module (Performance Monitoring Infrastructure) von WebSphere Application Server:** Verwenden Sie PMI, wenn Sie WebSphere eXtreme Scale in WebSphere Application Server ausführen. Diese Module liefern eine Sicht der internen Statistikstruktur.

Statistics-API

Wie bei einer Baumstruktur-Map gibt es einen entsprechenden Pfad und einen Schlüssel, um ein bestimmtes Modul abzurufen, bzw. in diesem Fall eine Differenzierungs- oder Aggregationsstufe. Angenommen, es gibt bereits einen Stammknoten in der Baumstruktur und die Statistiken werden für eine Map mit dem Namen ″payroll″ erfasst, die zu einem ObjectGrid mit dem Namen ″accounting″ gehört. Um beispielsweise auf das Modul für die Aggregations- bzw. Differenzierungsstufe einer Map zuzugreifen, können Sie einen Zeichenfolgebereich (String[]) der Pfade übergeben. In diesem Fall würde dieser ″String[] {root, ″accounting″, ″payroll″}″ lauten, da jede Zeichenfolge den Pfad des Knotens darstellt. Der Vorteil dieser Struktur ist der, dass ein Benutzer den Bereich für jeden Knoten im Pfad angeben und die Aggregationsstufe für diesen Knoten abrufen kann. Wenn Sie also ″String[] {root, ″accounting″}″ übergeben, erhalten Sie zwar Map-Statistiken, allerdings für das gesamte Grid ″accounting.″ Damit kann der Benutzer sowohl die Typen der zu überwachenden Statistiken als auch die für die Anwendung erforderliche Aggregationsstufe angeben.

PMI-Module von WebSphere Application Server

WebSphere eXtreme Scale enthält Statistikmodule, die mit WebSphere Application Server PMI verwendet werden können. Wenn ein Profil von WebSphere Application Server mit WebSphere eXtreme Scale erweitert wird, integrierten die Erweiterungsscripts die Module von WebSphere eXtreme Scale automatisch in die Konfigurationsdateien von WebSphere Application Server. Mit PMI können Sie Statistikmodule aktivieren und inaktivieren, Statistiken automatisch mit verschiedenen Differenzierungsstufen automatisch zusammenfassen und die Daten über den integrierten Tivoli Performance Viewer selbst in einem Graphen darstellen. Weitere Informationen hierzu finden Sie im Abschnitt „Leistung mit WebSphere Application Server PMI überwachen″ auf Seite 271.

Integration von Managed Beans (MBean) in Produkte anderer Anbieter

Die APIs und Managed Beans von eXtreme Scale sind so konzipiert, dass sie problemlos in Überwachungsanwendungen anderer Anbieter integriert werden können. JConsole und MC4J sind zwei Beispiele für schlanke JMX-Konsolen (Java Management Extensions), die zum Analysieren von Informationen über eine eXtreme-Scale-Topologie verwendet werden können. Sie können auch die programmgesteuerten APIs verwenden, um Adapterimplementierungen zu schreiben, mit

denen Momentaufnahmen der eXtreme-Scale-Leistung erstellt und die eXtreme-Scale-Leistung überwacht werden können. WebSphere eXtreme Scale enthält eine Musterüberwachungsanwendung, die sofort einsatzfähige Funktionen bietet und die als Schablone für das Schreiben erweiterter angepasster Überwachungsdienstprogramme verwendet werden kann.

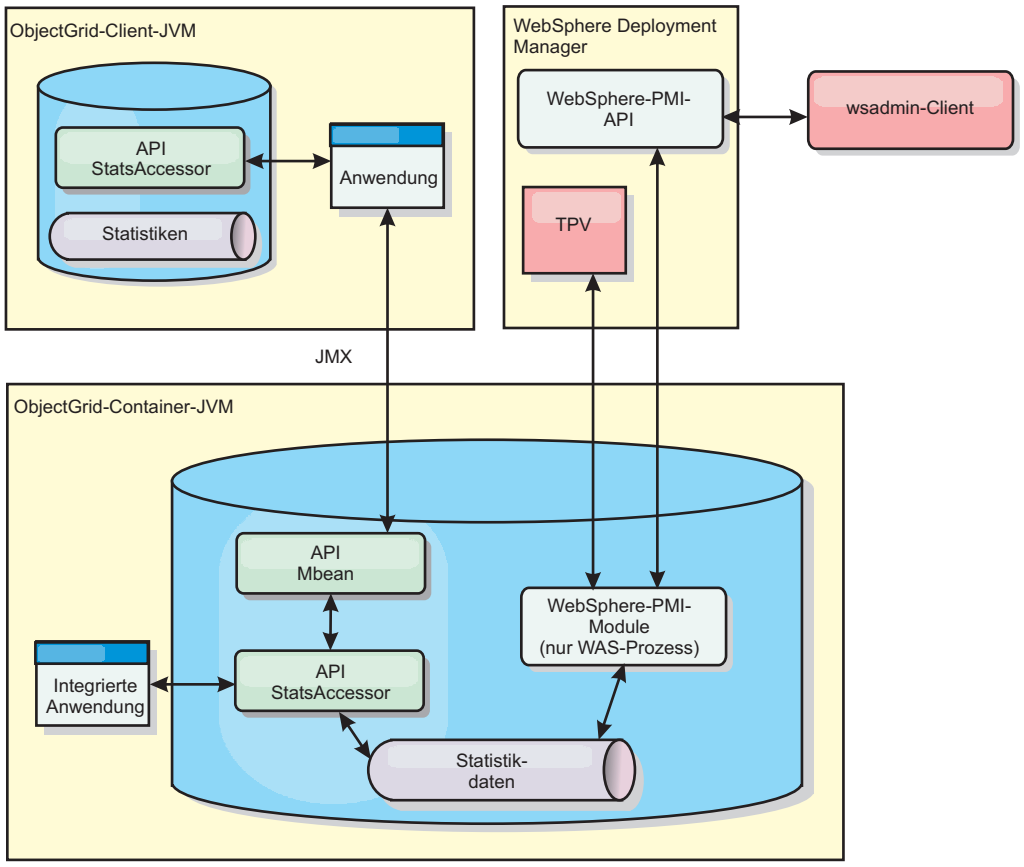


Abbildung 8. Übersicht über MBeans

Nähere Informationen hierzu finden Sie im Abschnitt „Musterdienstprogramm "xsAdmin" verwenden" auf Seite 287. Weitere Informationen zur Integration bestimmter Anwendungen anderer Anbieter finden Sie im folgenden Abschnitt:

- eXtreme Scale mit IBM Tivoli Monitoring Agent überwachen
- „eXtreme Scale mit Hyperic HQ überwachen" auf Seite 301
- „eXtreme-Scale-Anwendungen mit CA Wily Introscope überwachen" auf Seite 297

Zugehörige Tasks

Kapitel 8, „Implementierungsumgebung überwachen", auf Seite 263
Sie können Anwendungsprogrammierschnittstellen (API, Application Programming Interface), MBeans, Protokolle und Dienstprogramme verwenden, um die Leistung Ihrer Anwendungsumgebung zu überwachen.

„Überwachung mit der API ″Statistics″" auf Seite 267
Die API ″Statistics″ ist die direkte Schnittstelle zur internen Statistikstruktur. Statistiken sind standardmäßig inaktiviert, können aber über die Definition einer Schnittstelle ″StatsSpec″ aktiviert werden. Eine Schnittstelle ″StatsSpec″ definiert, wie WebSphere eXtreme Scale Statistiken überwachen soll.

„Leistung mit WebSphere Application Server PMI überwachen" auf Seite 271
WebSphere eXtreme Scale unterstützt Performance Monitoring Infrastructure (PMI), wenn Sie mit einem Anwendungsserver von WebSphere Application Server oder WebSphere Extended Deployment arbeiten. PMI erfasst Leistungsdaten zu Laufzeitanwendungen und stellt Schnittstellen bereit, über die externe Anwendungen für die Überwachung von Leistungsdaten unterstützt werden. Sie können die Administrationskonsole oder das Tool ″wsadmin″ verwenden, um auf Überwachungsdaten zuzugreifen.

„PMI aktivieren" auf Seite 272
Sie können WebSphere Application Server Performance Monitoring Infrastructure (PMI) verwenden, um Statistiken auf jeder Stufe zu aktivieren und zu inaktivieren. So können Sie beispielsweise die Statistik für die Cachetrefferrate einer bestimmten Map aktivieren, und die Statistik für die Eintragsanzahl oder die Statistik für die Loader-Aktualisierungszeiten im Stapelbetrieb inaktivieren. PMI kann über die Administrationskonsole oder mit Scripting aktiviert werden.

„PMI-Statistiken abrufen" auf Seite 274
Wenn Sie PMI-Statistiken abrufen, können Sie sich einen Eindruck über die Leistung Ihrer eXtreme-Scale-Anwendungen verschaffen.

„Musterdienstprogramm ″xsAdmin″ verwenden" auf Seite 287
Mit dem Musterdienstprogramm ″xsAdmin″ können Sie Textinformationen zu Ihrer WebSphere eXtreme Scale-Topologie formatieren und anzeigen. Das Musterdienstprogramm stellt eine Methode für die Syntaxanalyse und die Erkennung aktuelle Implementierungsdaten bereit und kann als Grundlage für das Schreiben angepasster Dienstprogramme verwendet werden.

„eXtreme Scale mit Hyperic HQ überwachen" auf Seite 301
Hyperic HQ ist eine Überwachungslösung eines anderen Anbieters, die kostenlos als Open-Source-Lösung oder als Unternehmensprodukt verfügbar ist. WebSphere eXtreme Scale enthält ein Plug-in, mit dem Hyperic-HQ-Agenten eXtreme-Scale-Containerserver erkennen und Statistiken mit Hilfe von eXtreme-Scale-Management-Beans berichten und zusammenfassen können. Sie können Hyperic HQ verwenden, um eigenständige Implementierungen von eXtreme Scale zu überwachen.

„Überwachung mit IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale" auf Seite 291
IBM Tivoli Enterprise Monitoring Agent ist eine mit vielen Funktionen ausgestattete Überwachungslösung, die Sie verwenden können, um Datenbanken, Betriebssysteme und Server in verteilten Umgebungen und in Hostumgebungen zu überwachen. WebSphere eXtreme Scale enthält einen angepassten Agenten, den Sie verwenden können, um eXtreme-Scale-Management-Beans selbst zu überwachen. Diese Lösung funktioniert effizient in eigenständigen Implementierungen von eXtreme Scale und in Implementierungen von eXtreme Scale mit WebSphere Application Server.

Zugehörige Verweise

„Managed Beans (MBeans) für die Verwaltung der Umgebung verwenden" auf Seite 285

Seite 285
Sie können verschiedene Typen von JMX-Beans (Java Management Extensions) verwenden, um Implementierungen zu verwalten und zu überwachen. Jede MBean bezieht sich auf eine bestimmte Entität, z. B. eine Map, eXtreme Scale, einen Server, eine Replikationsgruppe oder ein Replikationsgruppen-Member.

„PMI-Module" auf Seite 276
Sie können die Leistung Ihrer Anwendungen mit den PMI-Modulen (Performance Monitoring Infrastructure) überwachen.

„PMI-Module" auf Seite 276
Sie können die Leistung Ihrer Anwendungen mit den PMI-Modulen (Performance Monitoring Infrastructure) überwachen.

Überwachung mit der API "Statistics"

Die API "Statistics" ist die direkte Schnittstelle zur internen Statistikstruktur. Statistiken sind standardmäßig inaktiviert, können aber über die Definition einer Schnittstelle "StatsSpec" aktiviert werden. Eine Schnittstelle "StatsSpec" definiert, wie WebSphere eXtreme Scale Statistiken überwachen soll.

Warum und wann dieser Vorgang ausgeführt wird

Sie können die lokale API "StatsAccessor" verwenden, um Daten abzufragen und auf Statistiken zu jeder ObjectGrid-Instanz zuzugreifen, die in derselben Java Virtual Machine (JVM) wie der aktive Code ausgeführt wird. Weitere Informationen zu den einzelnen Schnittstellen finden Sie in der API-Dokumentation. Verwenden Sie die folgenden Schritte, um die Überwachung der internen Statistikstruktur zu aktivieren.

1. Rufen Sie das StatsAccessor-Objekt ab. Die Schnittstelle "StatsAccessor" folgt dem Singleton-Muster. Abgesehen von Problemen mit dem Klassenladeprogramm sollte deshalb nur eine einzige StatsAccessor-Instanz für jede JVM vorhanden sein. Diese Klasse dient als Hauptschnittstelle für alle lokalen Statistikoperationen. Der folgende Beispielcode veranschaulicht, wie die Klasse "accessor" abgerufen wird. Rufen Sie diese Operation auf, bevor Sie andere ObjectGrid-Aufrufe absetzen.

```
public class LocalClient {

   public static void main(String[] args) {

      // Handle für StatsAccessor abrufen
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

   }

}
```

2. Definieren Sie die Schnittstelle "StatsSpec" für das Grid. Definieren Sie diese JVM so, dass alle Statistiken nur auf ObjectGrid-Ebene erfasst werden. Sie müssen sicherstellen, dass eine Anwendung alle Statistiken aktiviert, die möglicherweise erforderlich sind, bevor Sie Transaktionen starten. Im folgenden Beispiel wird die Schnittstelle "StatsSpec" mit einem statischen Konstantenfeld und einer Spezifikationszeichenfolge definiert. Die Verwendung eines statischen Konstantenfelds ist einfacher, weil das Feld bereits die Spezifikation definiert hat. Wenn Sie jedoch eine Spezifikationszeichenfolge verwenden, können Sie jede erforderliche Kombination von Statistiken aktivieren.

```
public static void main(String[] args) {

      // Handle für StatsAccessor abrufen
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Spezifikation über das statische Feld definieren.
      StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);
      accessor.setStatsSpec(spec);
```

```
      // Spezifikation über die Spezifikationszeichenfolge definieren.
      StatsSpec spec = new StatsSpec("og.all=enabled");
      accessor.setStatsSpec(spec);

}
```

3. Senden Sie Transaktionen an das Grid, damit Daten für die Überwachung erfasst werden. Zum Erfassen hilfreicher Daten für Statistiken müssen Sie Transaktionen an das Grid senden. Der folgende Codeauszug fügt einen Datensatz in MapA ein, die in ObjectGridA enthalten ist. Da die Statistiken auf ObjectGrid-Ebene erfasst werden, liefert jede Map im ObjectGrid dieselben Ergebnisse.

```
public static void main(String[] args) {

      // Handle für StatsAccessor abrufen
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Spezifikation über das statische Feld definieren.
      StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);
      accessor.setStatsSpec(spec);

      ObjectGridManager manager =
    ObjectGridmanagerFactory.getObjectGridManager();
      ObjectGrid grid = manager.getObjectGrid("ObjectGridA");
      Session session = grid.getSession();
      Map map = session.getMap("MapA");

      // Einfügenvorgang starten.
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();
}
```

4. Fragen Sie eine StatsFact mit der API ″StatsAccessor ab. Jedem Statistikpfad wird eine Schnittstelle ″StatsFact″ zugeordnet. Die Schnittstelle ″StatsFact″ ist ein generischer Platzhalter, der verwendet wird, um ein StatsModule-Objekt zu organisieren und aufzunehmen. Bevor Sie auf das eigentliche Statistikmodul zugreifen können, muss das StatsFact-Objekt abgerufen werden.

```
public static void main(String[] args) {

      // Handle für StatsAccessor abrufen
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Spezifikation über das statische Feld definieren.
      StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);
      accessor.setStatsSpec(spec);

      ObjectGridManager manager =
    ObjectGridManagerFactory.getObjectGridManager();
      ObjectGrid grid = manager.getObjectGrid("ObjectGridA");
      Session session = grid.getSession();
      Map map = session.getMap("MapA");

      // Einfügenvorgang starten.
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();

      // StatsFact abrufen

      StatsFact fact = accessor.getStatsFact(new String[] {"EmployeeGrid"},
    StatsModule.MODULE_TYPE_OBJECT_GRID);

}
```

5. Interagieren Sie mit dem StatsModule-Objekt. Das StatsModule-Objekt ist in der Schnittstelle ″StatsFact″ enthalten. Sie können eine Referenz auf das Modul über die Schnittstelle ″StatsFact″ anfordern. Da die Schnittstelle ″StatsFact″ eine generische Schnittstelle ist, müssen Sie das zurückgegebene Modul in den erwarteten StatsModule-Typ umsetzen. Da diese Task eXtreme-Scale-Statistiken erfasst,

wird das zurückgegebene StatsModule-Objekt in den Typ ″OGStatsModule″ umgesetzt. Nach der Umsetzung des Moduls haben Sie Zugriff auf alle verfügbaren Statistiken.

```
public static void main(String[] args) {

      // Handle für StatsAccessor abrufen
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Spezifikation über das statische Feld definieren.
      StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);
      accessor.setStatsSpec(spec);

      ObjectGridManager manager =
    ObjectGridmanagerFactory.getObjectGridManager();
      ObjectGrid grid = manager.getObjectGrid("ObjectGridA");
      Session session = grid.getSession();
      Map map = session.getMap("MapA");

      // Einfügenvorgang starten.
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();

      // StatsFact abrufen
      StatsFact fact = accessor.getStatsFact(new String[] {"EmployeeGrid"},
    StatsModule.MODULE_TYPE_OBJECT_GRID);

      // Modul und Zeit abrufen
      OGStatsModule module = (OGStatsModule)fact.getStatsModule();
      ActiveTimeStatistic timeStat =
    module.getTransactionTime("Default", true);
      double time = timeStat.getMeanTime();

}
```

Zugehörige Konzepte

„Übersicht über Statistiken” auf Seite 263
Statistiken in WebSphere eXtreme Scale werden über eine interne Statistikstruktur erstellt. Die StatsAccessor-API, die PMI-Module (Performance Monitoring Infrastructure) und die MBean-API werden aus der internen Struktur erstellt.

„Tool eines anderen Anbieters” auf Seite 290
WebSphere eXtreme Scale kann mit verschiedenen gängigen Lösungen für die Unternehmensüberwachung überwacht werden. Es werden Plug-in-Agenten für IBM Tivoli Monitoring and Hyperic HQ bereitgestellt, die WebSphere eXtreme Scale mit Hilfe öffentlich zugänglicher Management-Beans überwachen. CA Wily Introscope verwendet die Java-Methodeninstrumentierung, um Statistiken zu erfassen.

„Statistikmodule”
WebSphere eXtreme Scale verwendet ein internes Statistikmodell, um Daten zu überwachen und zu filtern. Dieses Modell ist die grundlegende Struktur, die alle Datensichten verwenden, um Momentaufnahmen von Statistiken zu erfassen. Sie können verschiedene Methoden verwenden, um die Informationen von den Statistikmodulen abzurufen.

Zugehörige Verweise

„Managed Beans (MBeans) für die Verwaltung der Umgebung verwenden” auf Seite 285
Sie können verschiedene Typen von JMX-Beans (Java Management Extensions) verwenden, um Implementierungen zu verwalten und zu überwachen. Jede MBean bezieht sich auf eine bestimmte Entität, z. B. eine Map, eXtreme Scale, einen Server, eine Replikationsgruppe oder ein Replikationsgruppen-Member.

Statistikmodule

WebSphere eXtreme Scale verwendet ein internes Statistikmodell, um Daten zu überwachen und zu filtern. Dieses Modell ist die grundlegende Struktur, die alle

Datensichten verwenden, um Momentaufnahmen von Statistiken zu erfassen. Sie können verschiedene Methoden verwenden, um die Informationen von den Statistikmodulen abzurufen.

Übersicht

Statistiken in WebSphere eXtreme Scale werden überwacht und sind in StatsModules-Komponenten enthalten. Im Statistikmodell sind mehrere Typen von Statistikmodulen enthalten:

OGStatsModule
Liefert eine Statistik für eine ObjectGrid-Instanz, einschließlich der Transaktionsantwortzeiten.

MapStatsModule
Liefert eine Statistik für eine einzelne Map, einschließlich der Anzahl an Einträgen und der Trefferrate.

QueryStatsModule
Liefert eine Statistik zu Abfragen, einschließlich der Planerstellung und der Ausführungszeiten.

AgentStatsModule
Liefert eine Statistik zu DataGrid-API-Agenten, einschließlich Serialisierungs- und Ausführungszeiten.

HashIndexStatsModule
Liefert eine Statistik zu den Ausführungszeiten von HashIndex-Abfragen und Wartung.

SessionStatsModule
Liefert eine Statistik zum Plug-in für den HTTP-Sitzungsmanager.

Einzelheiten zu den Statistikmodulen finden Sie in in der Beschreibung des Pakets "com.ibm.websphere.objectgrid.stats" in der API-Dokumentation.

Statistiken in einer lokalen Umgebung

Das Modell ist wie eine n-stufige Baumstruktur organisiert, die sich aus allen in der vorherigen Liste erwähnten StatsModule-Typen zusammensetzt. Aufgrund dieser Organisationsstruktur wird jeder Knoten in der Baumstruktur durch die Schnittstelle "StatsFact" dargestellt. Die Schnittstelle "StatsFact" kann ein einzelnes Modul oder zu Aggregationszwecken eine Gruppe von Modulen darstellen. Wenn beispielsweise mehrere Blattknoten in der Baumstruktur bestimmte MapStatsModule-Objekte darstellen, enthält der übergeordnete StatsFact-Knoten dieser Knoten zusammengefasste Statistiken für alle untergeordneten Module. Nach dem Abruf eines StatsFact-Objekts können Sie die Schnittstelle zum Abrufen des entsprechenden StatsModule verwenden.

Wie in einer Baumstruktur-Map verwenden Sie einen entsprechenden Pfad oder einen Schlüssel, um ein bestimmtes StatsFact abzurufen. Der Pfad ist ein String[]-Wert, der sich aus allen Knoten im Pfad zum angeforderten Fakt zusammengesetzt. Beispiel: Sie haben ein ObjectGrid mit dem Namen "ObjectGridA" erstellt, das zwei Maps enthält: MapA und MapB. Der Pfad zum StatsModule für Map A ist `[ObjectGridA, MapA]`. Der Pfad zu den zusammengefassten Statistiken für beide Maps ist `[ObjectGridA]`.

Statistiken in einer verteilten Umgebung

In einer verteilten Umgebung werden die Statistikmodule über einen anderen Pfad abgerufen. Da ein Server mehrere Partitionen enthalten kann, muss die Statistikbaumstruktur die Partitionen überwachen, zu denen die einzelnen Module gehören. Deshalb ist der Suchpfad für ein bestimmtes StatsFact-Objekt anders. Wenn Sie das vorherige Beispiel erweitern und hinzufügen, dass sich die Maps in Partition 1 befinden, lautet der Pfad zum Abrufen des StatsFact-Objekts für Map A `[1, ObjectGridA, MapA]`.

Zugehörige Tasks

„Überwachung mit der API ″Statistics″" auf Seite 267
Die API ″Statistics″ ist die direkte Schnittstelle zur internen Statistikstruktur. Statistiken sind standardmäßig inaktiviert, können aber über die Definition einer Schnittstelle ″StatsSpec″ aktiviert werden. Eine Schnittstelle ″StatsSpec″ definiert, wie WebSphere eXtreme Scale Statistiken überwachen soll.

Leistung mit WebSphere Application Server PMI überwachen

WebSphere eXtreme Scale unterstützt Performance Monitoring Infrastructure (PMI), wenn Sie mit einem Anwendungsserver von WebSphere Application Server oder WebSphere Extended Deployment arbeiten. PMI erfasst Leistungsdaten zu Laufzeitanwendungen und stellt Schnittstellen bereit, über die externe Anwendungen für die Überwachung von Leistungsdaten unterstützt werden. Sie können die Administrationskonsole oder das Tool ″wsadmin″ verwenden, um auf Überwachungsdaten zuzugreifen.

Vorbereitungen

- Sie können PMI verwenden, um Ihre Umgebung zu überwachen, wenn Sie WebSphere eXtreme Scale in Kombination mit WebSphere Application Server verwenden.

Warum und wann dieser Vorgang ausgeführt wird

WebSphere eXtreme Scale verwendet das angepasste PMI-Feature von WebSphere Application Server, um eine eigene PMI-Instrumentierung hinzuzufügen. Über diesen Ansatz können Sie WebSphere eXtreme Scale PMI mit der Administrationskonsole oder mit JMX-Schnittstellen (Java Management Extensions) im Tool ″wsadmin″ aktivieren oder inaktivieren. Außerdem können Sie über die Standard-PMI- und -JMX-Schnittstellen, die von Überwachungsstools wie Tivoli Performance Viewer verwendet werden, auf Statistiken von WebSphere eXtreme Scale zugreifen.

1. Aktivieren Sie PMI in eXtreme Scale. Sie müssen PMI aktivieren, um die PMI-Statistiken anzeigen zu können. Nähere Informationen hierzu finden Sie im Abschnitt „PMI aktivieren" auf Seite 272.
2. Rufen Sie PMI-Statistiken von eXtreme Scale ab. Zeigen Sie die Leistung Ihrer eXtreme-Scale-Anwendungen mit Tivoli Performance Viewer an. Nähere Informationen hierzu finden Sie im Abschnitt „PMI-Statistiken abrufen" auf Seite 274.

Nächste Maßnahme

Weitere Informationen zum Tool ″wsadmin″ finden Sie im Abschnitt „Dienstprogramm ″wsadmin″ verwenden" auf Seite 284.

Zugehörige Konzepte

„Übersicht über Statistiken" auf Seite 263
Statistiken in WebSphere eXtreme Scale werden über eine interne Statistikstruktur erstellt. Die StatsAccessor-API, die PMI-Module (Performance Monitoring Infrastructure) und die MBean-API werden aus der internen Struktur erstellt.

„Tool eines anderen Anbieters" auf Seite 290
WebSphere eXtreme Scale kann mit verschiedenen gängigen Lösungen für die Unternehmensüberwachung überwacht werden. Es werden Plug-in-Agenten für IBM Tivoli Monitoring and Hyperic HQ bereitgestellt, die WebSphere eXtreme Scale mit Hilfe öffentlich zugänglicher Management-Beans überwachen. CA Wily Introscope verwendet die Java-Methodeninstrumentierung, um Statistiken zu erfassen.

Zugehörige Verweise

„Managed Beans (MBeans) für die Verwaltung der Umgebung verwenden" auf Seite 285
Sie können verschiedene Typen von JMX-Beans (Java Management Extensions) verwenden, um Implementierungen zu verwalten und zu überwachen. Jede MBean bezieht sich auf eine bestimmte Entität, z. B. eine Map, eXtreme Scale, einen Server, eine Replikationsgruppe oder ein Replikationsgruppen-Member.

„PMI-Module" auf Seite 276
Sie können die Leistung Ihrer Anwendungen mit den PMI-Modulen (Performance Monitoring Infrastructure) überwachen.

PMI aktivieren

Sie können WebSphere Application Server Performance Monitoring Infrastructure (PMI) verwenden, um Statistiken auf jeder Stufe zu aktivieren und zu inaktivieren. So können Sie beispielsweise die Statistik für die Cachetrefferrate einer bestimmten Map aktivieren, und die Statistik für die Eintragsanzahl oder die Statistik für die Loader-Aktualisierungszeiten im Stapelbetrieb inaktivieren. PMI kann über die Administrationskonsole oder mit Scripting aktiviert werden.

Vorbereitungen

Ihr Anwendungsserver muss gestartet sein und eine installierte Anwendung haben, die für eXtreme Scale aktiviert ist. Zum Aktivieren von PMI mit Scripting müssen Sie sich anmelden und das Tool "wsadmin" verwenden können. Weitere Informationen zum Tool "wsadmin" finden Sie im Abschnitt Tool "wsadmin" im Information Center von WebSphere Application Server.

Warum und wann dieser Vorgang ausgeführt wird

Verwenden Sie WebSphere Application Server PMI, um einen differenzierten Mechanismus bereitzustellen, mit dem Sie Statistiken auf jeder Stufe aktivieren und inaktivieren können. So können Sie beispielsweise die Statistik für die Cachetreffertrate einer bestimmten Map aktivieren, und die Statistik für die Eintragsanzahl oder die Statistik für die Loader-Aktualisierungszeiten im Stapelbetrieb inaktivieren. In diesem Abschnitt wird beschrieben, wie Sie PMI über die Administrationskonsole und mit wsadmin-Scripts für ObjectGrid aktivieren können.

- **PMI über die Administrationskonsole aktivieren**
  1. Klicken Sie in der Administrationskonsole auf **Überwachung und Optimierung → Performance Monitoring Infrastructure →** *Servername*.
  2. Stellen Sie sicher, dass "Performance Monitoring Infrastructure (PMI) aktivieren" ausgewählt ist. Diese Einstellung ist standardmäßig aktiviert. Wenn die

Einstellung nicht aktiviert ist, wählen Sie das Kontrollkästchen aus, und starten Sie anschließend den Server erneut.

3. Klicken Sie auf **Angepasst**. Wählen Sie in der Konfigurationsstruktur das ObjectGrid und das ObjectGrid-Modul "Maps" aus. Aktivieren Sie die Statistik für jedes Modul.

Die Transaktionstypkategorie für ObjectGrid-Statistiken wird zur Laufzeit erstellt. Die Unterkategorien der ObjectGrid- und Map-Statistiken sind nur auf der Registerkarte **Laufzeit** sichtbar.

- **PMI mit Scripting aktivieren**
  1. Öffnen Sie eine Befehlszeile. Navigieren Sie zum Verzeichnis "Installationsstammverzeichnis/bin". Geben Sie "wsadmin" ein, um das Befehlszeilentool "wsadmin" zu starten.
  2. Ändern Sie die PMI-Laufzeitkonfiguration für eXtreme Scale. Stellen Sie mit den folgenden Befehlen sicher, dass PMI für den Server aktiviert ist:

```
wsadmin>set s1 [$AdminConfig getid /Cell:ZELLENNAME/Node:KNOTENNAME/Server:ANWENDUNGSSERVERNAME/]
wsadmin>set pmi [$AdminConfig list PMIService $s1]
wsadmin>$AdminConfig show $pmi.
```

Wenn PMI nicht aktiviert ist, führen Sie die folgenden Befehle aus, um PMI zu aktivieren:

```
wsadmin>$AdminConfig modify $pmi {{enable true}}
wsadmin>$AdminConfig save
```

Wenn Sie PMI aktivieren müssen, starten Sie den Server erneut.

3. Setzen Sie unter Verwendung der folgenden Befehle Variablen, um die Statistikgruppe in eine angepasste Gruppe zu ändern:

```
wsadmin>set perfName [$AdminControl completeObjectName type=Perf,
process=ANWENDUNGSSERVERNAME,*]
wsadmin>set perfOName [$AdminControl makeObjectName $perfName]
wsadmin>set params [java::new {java.lang.Object[]} 1]
wsadmin>$params set 0 [java::new java.lang.String custom]
wsadmin>set sigs [java::new {java.lang.String[]} 1]
wsadmin>$sigs set 0 java.lang.String
```

4. Setzen Sie die Statistikgruppe mit dem folgenden Befehl auf "custom" (Angepasst):

```
wsadmin>$AdminControl invoke_jmx $perfOName setStatisticSet $params $sigs
```

5. Setzen Sie mit den folgenden Befehlen Variablen, um die PMI-Statistik "objectGridModule" zu aktivieren:

```
wsadmin>set params [java::new {java.lang.Object[]} 2]
wsadmin>$params set 0 [java::new java.lang.String objectGridModule=1]
wsadmin>$params set 1 [java::new java.lang.Boolean false]
wsadmin>set sigs [java::new {java.lang.String[]} 2]
wsadmin>$sigs set 0 java.lang.String
wsadmin>$sigs set 1 java.lang.Boolean
```

6. Setzen Sie die Statistikzeichenfolge mit dem folgenden Befehl:

```
wsadmin>set params2 [java::new {java.lang.Object[]} 2]
wsadmin>$params2 set 0 [java::new java.lang.String mapModule=*]
wsadmin>$params2 set 1 [java::new java.lang.Boolean false]
wsadmin>set sigs2 [java::new {java.lang.String[]} 2]
wsadmin>$sigs2 set 0 java.lang.String
wsadmin>$sigs2 set 1 java.lang.Boolean
```

7. Setzen Sie die Statistikzeichenfolge mit dem folgenden Befehl:

```
wsadmin>$AdminControl invoke_jmx $perfOName setCustomSetString $params2 $sigs2
```

Mit diesen Schritten haben Sie PMI für die Laufzeitumgebung von eXtreme Scale aktiviert, aber die PMI-Konfiguration nicht geändert. Wenn Sie den

Anwendungsserver erneut starten, gehen die PMI-Einstellungen bis auf die eigentliche Aktivierung von PMI verloren.

Beispiel

Sie können die folgenden Schritte ausführen, um die PMI-Statistiken für die Musteranwendung zu aktivieren:

1. Starten Sie die Anwendung mit der Webadresse `http://Host:Port/ObjectGridSample`, wobei "Host" und "Port" für den Hostnamen und die HTTP-Portnummer des Servers stehen, in dem die Musteranwendung installiert ist.
2. Klicken Sie in der Musteranwendung auf "ObjectGridCreationServlet" und anschließend auf die Aktionsschaltflächen 1, 2, 3, 4 und 5, um Aktionen für das ObjectGrid und die Maps zu generieren. Schließen Sie diese Servlet-Seite noch nicht.
3. Klicken Sie in der Administrationskonsole auf **Überwachung und Optimierung** → **Performance Monitoring Infrastructure** → *Servername*. Klicken Sie auf das Register **Laufzeit**.
4. Klicken Sie auf das Optionsfeld **Angepasst**.
5. Erweitern Sie das ObjectGrid-Modul "Maps" in der Laufzeitstruktur, und klicken Sie anschließend auf den Link "clusterObjectGrid". In der ObjectGrid-Gruppe "Maps" gibt es eine ObjectGrid-Instanz "clusterObjectGrid", und in der Gruppe "clusterObjectGrid" gibt es vier Maps: counters, employees, offices und sites. Die ObjectGrid-Instanz enthält die clusterObjectGrid-Instanz, und diese Instanz hat den Transaktionstyp DEFAULT.
6. Sie können die gewünschten Statistiken aktivieren. Sie können beispielsweise die Statistik für die Anzahl der Einträge in der Map "employees" und die Statistik für die Transaktionsantwortzeiten für den Transaktionstyp DEFAULT aktivieren.

Nächste Maßnahme

Nach der Aktivierung von PMI können Sie die PMI-Statistiken über die Administrationskonsole oder mit Scripting anzeigen.

Zugehörige Konzepte

„Übersicht über Statistiken" auf Seite 263
Statistiken in WebSphere eXtreme Scale werden über eine interne Statistikstruktur erstellt. Die StatsAccessor-API, die PMI-Module (Performance Monitoring Infrastructure) und die MBean-API werden aus der internen Struktur erstellt.

Zugehörige Verweise

„PMI-Module" auf Seite 276
Sie können die Leistung Ihrer Anwendungen mit den PMI-Modulen (Performance Monitoring Infrastructure) überwachen.

PMI-Statistiken abrufen

Wenn Sie PMI-Statistiken abrufen, können Sie sich einen Eindruck über die Leistung Ihrer eXtreme-Scale-Anwendungen verschaffen.

Vorbereitungen

- Aktivieren Sie die Verfolgung von PMI-Statistiken für Ihre Umgebung. Weitere Informationen hierzu finden Sie im Abschnitt „PMI aktivieren" auf Seite 272.

- Bei den Pfadangaben in dieser Task wird davon ausgegangen, dass Sie Statistiken für die Musterandung abrufen, aber Sie können diese Statistiken mit ähnlichen Schritten für jede andere Anwendung verwenden.
- Wenn Sie die Administrationskonsole verwenden, um Statistiken abzurufen, müssen Sie sich an der Administrationskonsole anmelden können. Wenn Sie Scripting verwenden, müssen Sie sich bei wsadmin anmelden können.

Warum und wann dieser Vorgang ausgeführt wird

Sie können PMI-Statistiken abrufen und diese in Tivoli Performance Viewer anzeigen, indem Sie Schritte in der Administrationskonsole oder mit Scripting ausführen.

- Schritte für die Administrationskonsole
- Schritte für Scripting

Weitere Informationen zu den Statistiken, die abgerufen werden können, finden Sie im Abschnitt „PMI-Module" auf Seite 276.

- Rufen Sie PMI-Statistiken in der Administrationskonsole ab.
  1. Klicken Sie in der Administrationskonsole auf **Überwachung und Optimierung** → **Performance Viewer** → **Aktuelle Aktivität**.
  2. Wählen Sie den Server, den Sie mit Tivoli Performance Viewer überwachen möchten, aus, und aktivieren Sie anschließend die Überwachung.
  3. Klicken Sie auf den Server, um die Seite "Performance Viewer" anzuzeigen.
  4. Erweitern Sie die Konfigurationsstruktur. Klicken Sie auf **ObjectGrid-Maps** → **clusterObjectGrid**, und wählen Sie **employees** aus. Klicken Sie auf **ObjectGrids** → **clusterObjectGrid**, und wählen Sie **DEFAULT** aus.
  5. Navigieren Sie in der ObjectGrid-Musteranwendung zum Servlet "ObjectGridCreationServlet", klicken Sie auf Schaltfläche 1, und füllen Sie die Maps mit Daten. Sie können die Statistiken im Viewer anzeigen.
- Rufen Sie PMI-Statistiken mit Scripting ab.
  1. Navigieren Sie über eine Befehlszeile zum Verzeichnis `Installationsstammverzeichnis/bin`. Geben Sie `wsadmin` ein, um das Tool "wsadmin" zu starten.
  2. Setzen Sie mit den folgenden Befehlen Variablen für die Umgebung:

     ```
     wsadmin>set perfName [$AdminControl completeObjectName type=Perf,*]
     wsadmin>set perfOName [$AdminControl makeObjectName $perfName]
     wsadmin>set mySrvName [$AdminControl completeObjectName type=Server,
      name=APPLICATION_SERVER_NAME,*]
     ```

  3. Setzen Sie mit den folgenden Befehlen Variablen, um die Statistik "mapModule" abzurufen:

     ```
     wsadmin>set params [java::new {java.lang.Object[]} 3]
     wsadmin>$params set 0 [$AdminControl makeObjectName $mySrvName]
     wsadmin>$params set 1 [java::new java.lang.String mapModule]
     wsadmin>$params set 2 [java::new java.lang.Boolean true]
     wsadmin>set sigs [java::new {java.lang.String[]} 3]
     wsadmin>$sigs set 0 javax.management.ObjectName
     wsadmin>$sigs set 1 java.lang.String
     wsadmin>$sigs set 2 java.lang.Boolean
     ```

  4. Rufen Sie die Statistik "mapModule" mit dem folgenden Befehl ab:

     ```
     wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params $sigs
     ```

  5. Setzen Sie mit den folgenden Befehlen Variablen, um die Statistik "objectGridModule" abzurufen:

```
wsadmin>set params2 [java::new {java.lang.Object[]} 3]
wsadmin>$params2 set 0 [$AdminControl makeObjectName $mySrvName]
wsadmin>$params2 set 1 [java::new java.lang.String objectGridModule]
wsadmin>$params2 set 2 [java::new java.lang.Boolean true]
wsadmin>set sigs2 [java::new {java.lang.String[]} 3]
wsadmin>$sigs2 set 0 javax.management.ObjectName
wsadmin>$sigs2 set 1 java.lang.String
wsadmin>$sigs2 set 2 java.lang.Boolean
```

6. Rufen Sie die Statistik "objectGridModule" mit dem folgenden Befehl ab:

```
wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params2 $sigs2
```

Ergebnisse

Sie können Statistiken in Tivoli Performance Viewer anzeigen.

Zugehörige Konzepte

„Übersicht über Statistiken" auf Seite 263
Statistiken in WebSphere eXtreme Scale werden über eine interne Statistikstruktur erstellt. Die StatsAccessor-API, die PMI-Module (Performance Monitoring Infrastructure) und die MBean-API werden aus der internen Struktur erstellt.

Zugehörige Verweise

„PMI-Module"
Sie können die Leistung Ihrer Anwendungen mit den PMI-Modulen (Performance Monitoring Infrastructure) überwachen.

PMI-Module

Sie können die Leistung Ihrer Anwendungen mit den PMI-Modulen (Performance Monitoring Infrastructure) überwachen.

objectGridModule

Das Modul "objectGridModule" enthält eine Zeitstatistik: Antwortzeit der Transaktion. Eine Transaktion ist wie folgt definiert: Dauer zwischen dem Aufruf der Methode "Session.begin" und dem Aufruf der Methode "Session.commit". Diese Dauer wird als Antwortzeit der Transaktion verfolgt. Das Stammelement ("root") des Moduls "the objectGridModule" dient als Einstiegspunkt für die Statistiken von WebSphere eXtreme Scale. Dieses Stammelement hat ObjectGrids als untergeordnete Elemente, die dieselben Transaktionstypen wie deren untergeordnete Elemente haben. Die Antwortzeitstatistik wird jedem Transaktionstyp zugeordnet.

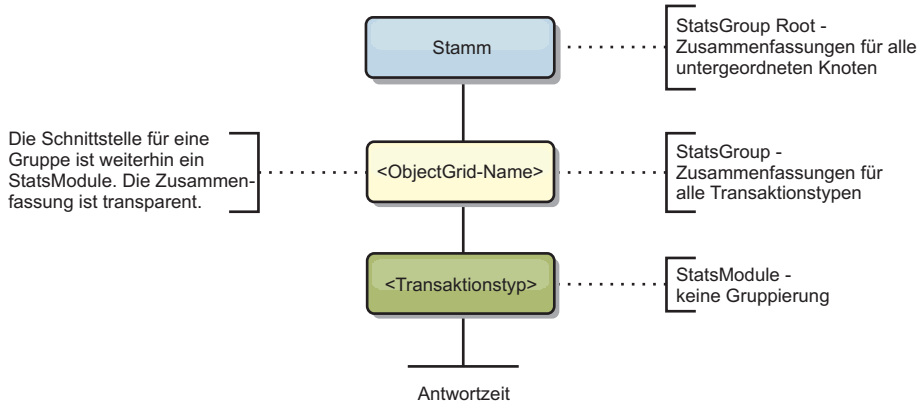


Abbildung 9. Struktur des Moduls "ObjectGridModule"

Die folgende Abbildung zeigt eine ObjectGridModule-Beispielstruktur. In diesem Beispiel sind zwei ObjectGrid-Instanzen im System vorhanden: ObjectGrid A und ObjectGrid B. Die ObjectGrid-Instanz A hat zwei Typen von Transaktionen: A und Standard. Die ObjectGrid-Instanz B hat nur den Transaktionstyp "Standard".

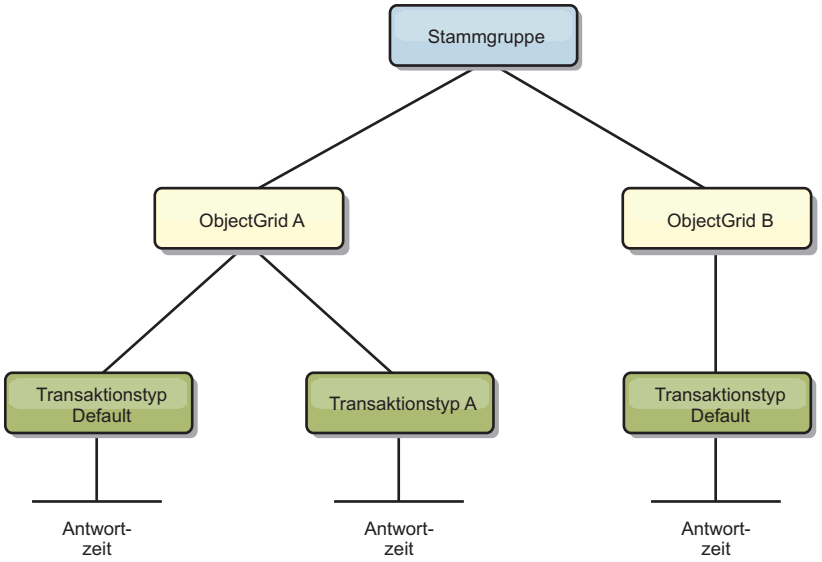


Abbildung 10. Beispielstruktur für das Modul "ObjectGridModule"

Transaktionstypen werden von Anwendungsentwicklern definiert, weil sie wissen, welche Typen von Transaktionen in ihren Anwendungen verwendet werden. Der Transaktionstyp wird mit der folgenden Methode "Session.setTransactionType(String)" gesetzt:

```
/**
* Legt den Transaktionstyp für künftige Transaktionen fest.
*
* Nach dem Aufruf dieser Methode haben alle künftigen Transaktionen denselben
* Typ, bis ein anderer Transaktionstyp festgelegt wird. Wenn Sie keinen
* Transaktionstyp festlegen, wird der Standardtransaktionstyp TRANSACTION_TYPE_DEFAULT
* verwendet.
*
* Transaktionstypen werden hauptsächlich für die Verfolgung statistischer Daten verwendet.
* Benutzer können Typen von Transaktionen, die in einer Anwendung ausgeführt werden,
* vordefinieren. Die Idee ist, Transaktionen mit denselben Merkmalen in einer
* Kategorie (Typ) zusammenzufassen, so dass jeweils eine einzige Statistik zur
* Transaktionsantwortzeit verwendet werden kann, um den jeweiligen Transaktionstyp zu verfolgen.
*
* Diese Verfolgung ist hilfreich, wenn Ihre Anwendung unterschiedliche
* Transaktionstypen hat.
* Einige Typen von Transaktionen, wie z. B. Aktualisierungstransaktionen,
* haben eine längere Verarbeitungszeit als andere Transaktionen, wie z. B.
* Lesetransaktionen. Wenn Sie den Transaktionstyp verwenden, können unterschiedliche
* Transaktionen über unterschiedliche Statistiken verfolgt werden, so dass
* die Statistiken hilfreicher sind.
*
* @param tranType Der Transaktionstyp für künftige Transaktionen.
*/
void setTransactionType(String tranType);
```

Im folgenden Beispiel wird der Transaktionstyp auf `updatePrice` gesetzt:

```
// Transaktionstyp auf "updatePrice" setzen.
// Die Zeit zwischen session.begin() und session.commit() wird in der
// Zeitstatistik für "updatePrice" verfolgt.
session.setTransactionType("updatePrice");
session.begin();
map.update(stockId, new Integer(100));
session.commit();
```

Die erste Zeile gibt an, dass der folgende Transaktionstyp "updatePrice" ist. In dem Beispiel ist eine Statistik "updatePrice" in der ObjectGrid-Instanz vorhanden, die dem Session-Objekt entspricht. Mit JMX-Schnittstellen (Java Management Extensi-

ons) können Sie die Transaktionsantwortzeit für Transaktionen des Typs ″updatePrice″ abrufen. Sie können auch die zusammengefasste Statistik für alle Transaktionstypen in der angegebenen ObjectGrid-Instanz abrufen.

mapModule

Das Modul ″mapModule″ enthält drei Statistiken, die sich auf eXtreme-Scale-Maps beziehen:

- **Map hit rate** - *BoundedRangeStatistic*: Verfolgt die Trefferrate einer Map. Die Trefferrate ist ein variabler Wert zwischen 0 und 100 einschließlich, der den Prozentsatz der Map-Treffer in Relation zu den get-Operationen für die Map darstellt.
- **Number of entries**-*CountStatistic*: Verfolgt die Anzahl der Einträge in der Map.
- **Loader batch update response time**-*TimeStatistic*: Verfolgt die Antwortzeit für Aktualisierungsoperationen im Stapelbetrieb des Loaders.

Das Stammelement (″root″) des Moduls ″mapModule″ dient als Einstiegspunkt für die ObjectGrid-Map-Statistiken. Dieses Stammelement hat Maps als untergeordnete Elemente, die dieselben Transaktionstypen wie deren untergeordnete Elemente haben. Jede Map-Instanz hat die drei aufgelisteten Statistiken. Die Struktur des Moduls ″mapModule″ ist in der folgenden Abbildung dargestellt:

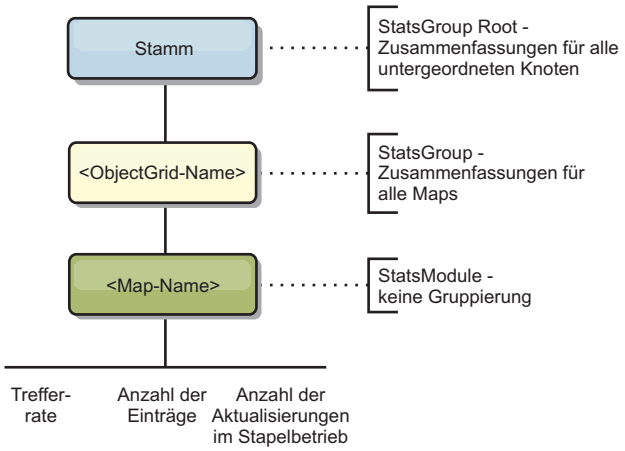
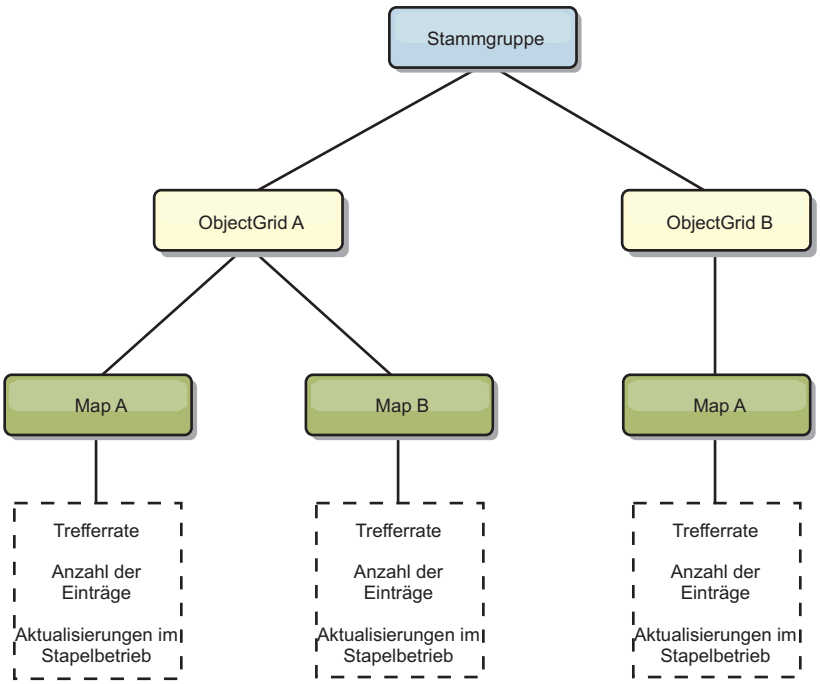


Abbildung 11. Struktur des Moduls ″mapModule″

Die folgende Abbildung zeigt eine Beispielstruktur für das Modul ″mapModule″:

Abbildung 12. Beispielstruktur für das Modul ″mapModule″



Modul "hashIndexModule"

Das Modul "hashIndexModule" enthält die folgenden Statistiken, die sich auf Indizes auf Map-Ebene beziehen:

- **Find Count**-*CountStatistic*: Die Anzahl der Aufrufe für die Indexoperation "find".
- **Collision Count**-*CountStatistic*: Die Anzahl der Kollisionen für die Operation "find".
- **Failure Count**-*CountStatistic*: Die Anzahl der Fehler für die Operation "find".
- **Result Count**-*CountStatistic*: Die Anzahl der von der Operation "find" zurückgegebenen Schlüssel.
- **BatchUpdate Count**-*CountStatistic*: Die Anzahl der für diesen Index ausgeführten Aktualisierungen im Stapelbetrieb. Wenn die entsprechende Map geändert wird, wird die Methode "doBatchUpdate()" des Index aufgerufen. Diese Statistik teilt Ihnen mit, wie oft sich der Index ändert bzw. aktualisiert wird.
- **Find Operation Duration Time**-*TimeStatistic*: Ausführungsdauer der Operation "find".

Das Stammelement ("root") des Moduls "hashIndexModule" "root" dient als Einstiegspunkt für die HashIndex-Statistik. Das Stammelement hat ObjectGrids als untergeordnete Elemente, ObjectGrids haben Maps als untergeordnete Elemente, die wiederum HashIndex-Instanzen als untergeordnete Elemente und Blattknoten der Baumstruktur haben. Jede HashIndex-Instanz hat die drei aufgelisteten Statistiken. Die Struktur des Moduls "hashIndexModule" wird in der folgenden Abbildung gezeigt:

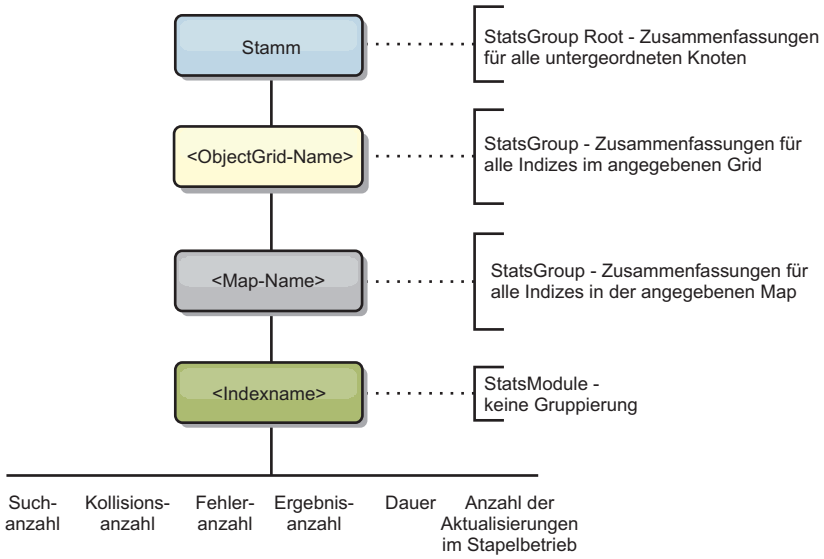


Abbildung 13. Struktur des Moduls "hashIndexModule"

Die folgende Abbildung zeigt eine Beispielstruktur für das Modul "hashIndexModule":

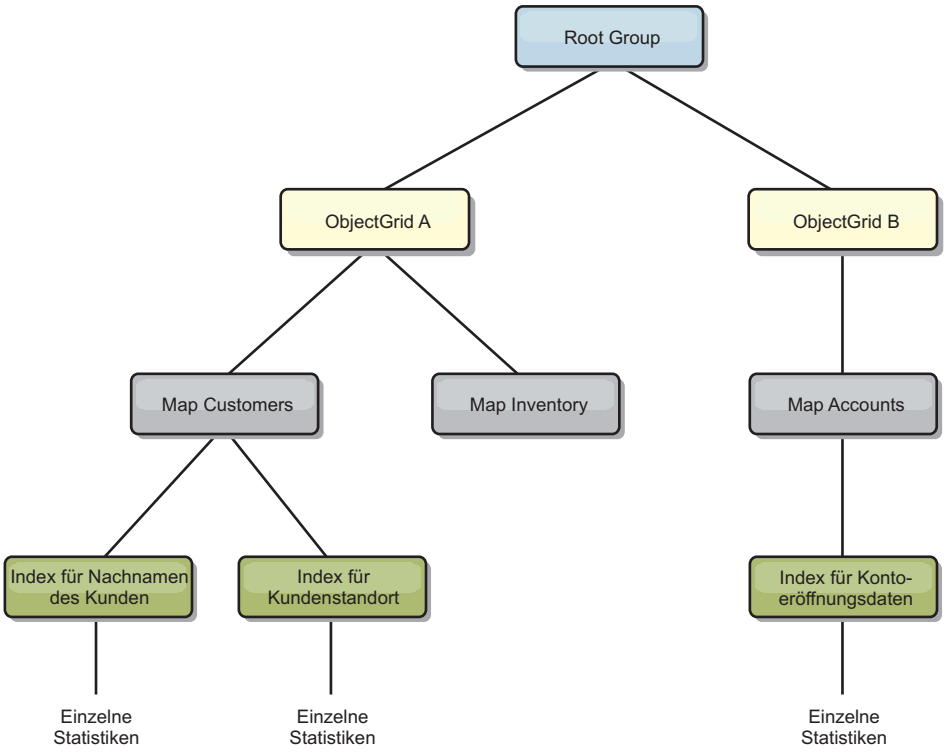


Abbildung 14. Beispielstruktur für das Modul "hashIndexModule"

Modul "agentManagerModule"

Das Modul "agentManagerModule" enthält Statistiken, die sich auf die Agenten auf Map-Ebene beziehen:

- **Reduce Time**: *TimeStatistic* - Die Zeit, die der Agent für die Ausführung der Operation "reduce" benötigt.

- **Total Duration Time**: *TimeStatistic* - Die Gesamtzeit, die der Agent für die Ausführung aller Operationen benötigt.
- **Agent Serialization Time**: *TimeStatistic* - Die Zeit für die Serialisierung des Agenten.
- **Agent Inflation Time**: *TimeStatistic* - Die Zeit, die benötigt wird, um den Agenten im Server zu dekomprimieren.
- **Result Serialization Time**: *TimeStatistic* - Die Zeit für die Serialisierung der Ergebnisse des Agenten.
- **Result Inflation Time**: *TimeStatistic* - Die Zeit für die Dekomprimierung der Ergebnisse des Agenten.
- **Failure Count**: *CountStatistic* - Die Anzahl der Fehler des Agenten.
- **Invocation Count**: *CountStatistic* - Die Anzahl der AgentManager-Aufrufe.
- **Partition Count**: *CountStatistic* - Die Anzahl der Partitionen, an die der Agent gesendet wird.

Das Stammelement ("root" des Moduls "agentManagerModule") dient als Einstiegspunkt für die AgentManager-Statistiken. Dieses Stammelement hat ObjectGrids als untergeordnete Elemente, die ObjectGrids haben Maps als untergeordnete Elemente, die wiederum AgentManager-Instanzen als untergeordnete Elemente und Blattknoten der Baumstruktur haben. Jede AgentManager-Instanz hat die drei aufgelisteten Statistiken.

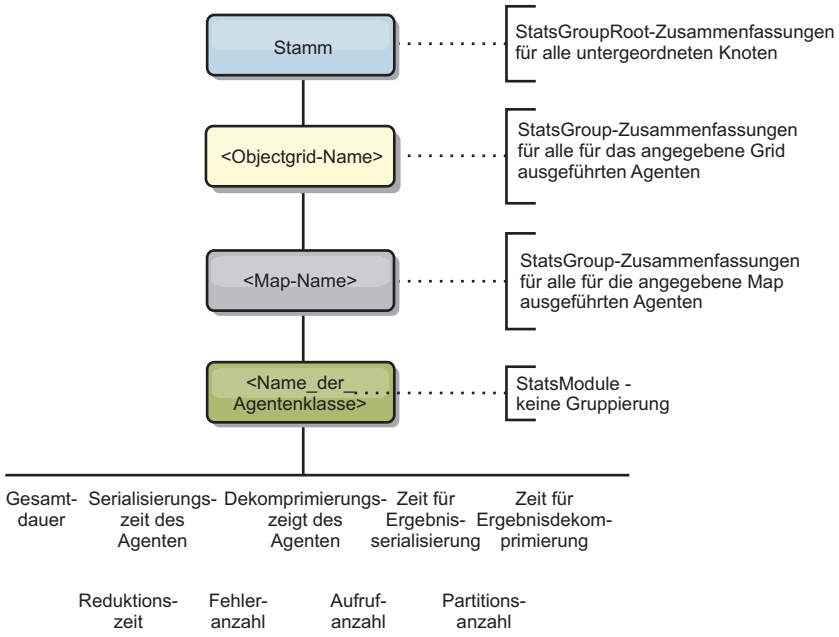


Abbildung 15. Struktur des Moduls "agentManagerModule"

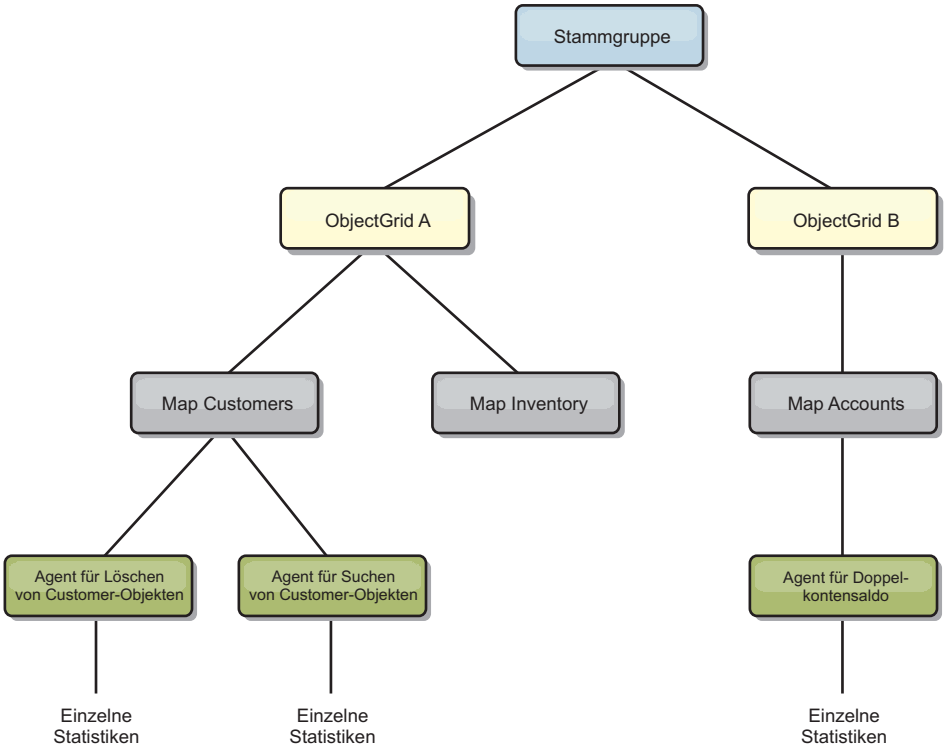


Abbildung 16. Beispielstruktur für das Modul "agentManagerModule"

Modul "queryModule"

Das Modul "queryModule" enthält Statistiken, die sich auf eXtreme-Scale-Abfragen beziehen:

- **Plan Creation Time**: *TimeStatistic* - Die Zeit für die Erstellung des Abfrageplans.
- **Execution Time**: *TimeStatistic* - Die Zeit für die Ausführung der Abfrage.
- **Execution Count**: *CountStatistic* - Die Anzahl der Abfrageläufe.
- **Result Count**: *CountStatistic* - Der Zähler für jede Ergebnismenge jedes Abfragelaufs.
- **FailureCount**: *CountStatistic* - Die Anzahl der Abfragefehler.

Das Stammelement ("root") des Moduls "queryModule" dient als Einstiegspunkt für die Abfragestatistiken. Dieses Stammelement hat ObjectGrids als untergeordnete Elemente, die Query-Objekt als untergeordnete Elemente und Blattknoten der Baumstruktur haben. Jede Query-Instanz hat die drei aufgelisteten Statistiken.

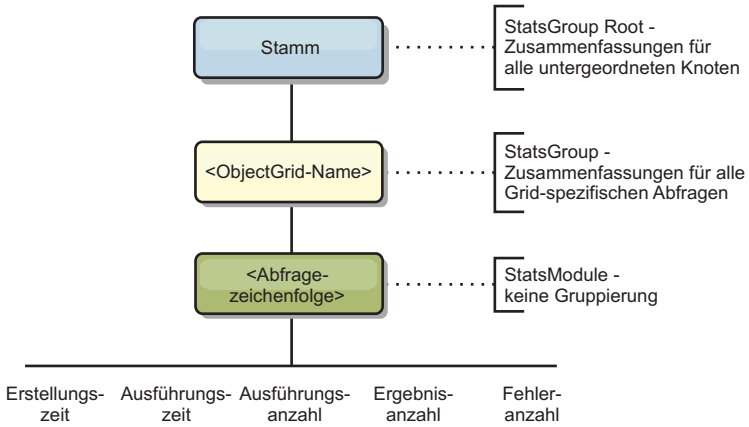


Abbildung 17. Struktur des Moduls "queryModule"

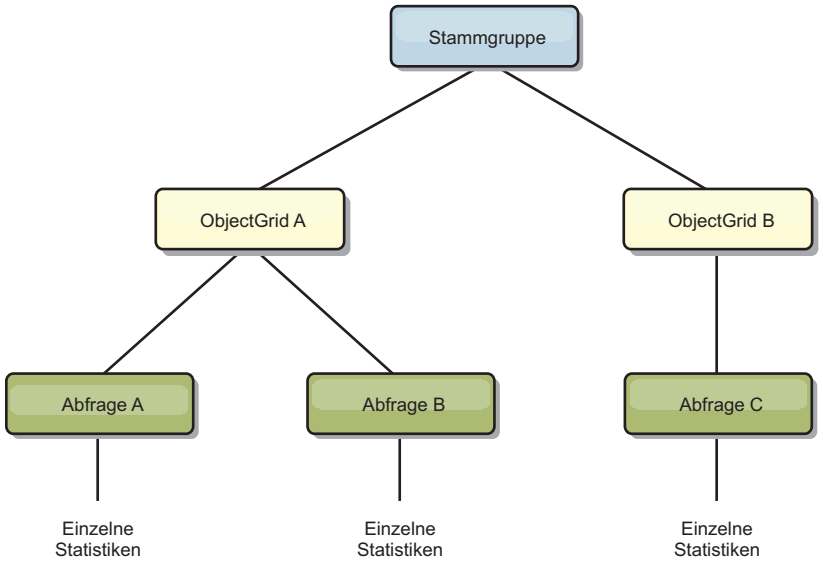


Abbildung 18. Beispielstruktur für das Modul "queryModule"

Zugehörige Konzepte

„Übersicht über Statistiken" auf Seite 263
Statistiken in WebSphere eXtreme Scale werden über eine interne Statistikstruktur erstellt. Die StatsAccessor-API, die PMI-Module (Performance Monitoring Infrastructure) und die MBean-API werden aus der internen Struktur erstellt.

Zugehörige Tasks

„Leistung mit WebSphere Application Server PMI überwachen" auf Seite 271
WebSphere eXtreme Scale unterstützt Performance Monitoring Infrastructure (PMI), wenn Sie mit einem Anwendungsserver von WebSphere Application Server oder WebSphere Extended Deployment arbeiten. PMI erfasst Leistungsdaten zu Laufzeitanwendungen und stellt Schnittstellen bereit, über die externe Anwendungen für die Überwachung von Leistungsdaten unterstützt werden. Sie können die Administrationskonsole oder das Tool ″wsadmin″ verwenden, um auf Überwachungsdaten zuzugreifen.

„PMI aktivieren" auf Seite 272
Sie können WebSphere Application Server Performance Monitoring Infrastructure (PMI) verwenden, um Statistiken auf jeder Stufe zu aktivieren und zu inaktivieren. So können Sie beispielsweise die Statistik für die Cachetrefferrate einer bestimmten Map aktivieren, und die Statistik für die Eintragsanzahl oder die Statistik für die Loader-Aktualisierungszeiten im Stapelbetrieb inaktivieren. PMI kann über die Administrationskonsole oder mit Scripting aktiviert werden.

„PMI-Statistiken abrufen" auf Seite 274
Wenn Sie PMI-Statistiken abrufen, können Sie sich einen Eindruck über die Leistung Ihrer eXtreme-Scale-Anwendungen verschaffen.

„Musterdienstprogramm ″xsAdmin″ verwenden" auf Seite 287
Mit dem Musterdienstprogramm ″xsAdmin″ können Sie Textinformationen zu Ihrer WebSphere eXtreme Scale-Topologie formatieren und anzeigen. Das Musterdienstprogramm stellt eine Methode für die Syntaxanalyse und die Erkennung aktuelle Implementierungsdaten bereit und kann als Grundlage für das Schreiben angepasster Dienstprogramme verwendet werden.

Dienstprogramm ″wsadmin″ verwenden

Sie können das in WebSphere Application Server bereitgestellte Dienstprogramm ″wsadmin″ verwenden, um auf die MBean-Informationen zuzugreifen.

Mit dem Tool ″wsadmin″ auf MBeans zugreifen

Führen Sie das Tool ″wsadmin″ im Verzeichnis ″bin″ Ihrer Installation von WebSphere Application Server auf. Im folgenden Beispiel wird eine Sicht der aktuellen Shard-Verteilung in einer dynamischen eXtreme-Scale-Umgebung abgerufen. Sie können wsadmin in jeder Installation ausführen, in der eXtreme Scale ausgeführt wird. Das Dienstprogramm ″wsadmin″ muss nicht im Katalogservice ausgeführt werden.

```
$ wsadmin.sh -lang jython
wsadmin>placementService = AdminControl.queryNames
 ("com.ibm.websphere.objectgrid:*,type=PlacementService")
wsadmin>print AdminControl.invoke(placementService,
 "listObjectGridPlacement","library ms1")

<objectGrid name="library" mapSetName="ms1">
  <container name="container-0" zoneName="DefaultDomain"
  hostName="host1.company.org" serverName="server1">
     <shard type="Primary" partitionName="0"/>
     <shard type="SynchronousReplica" partitionName="1"/>
  </container>
  <container name="container-1" zoneName="DefaultDomain"
  hostName="host2.company.org" serverName="server2">
     <shard type="SynchronousReplica" partitionName="0"/>
     <shard type="Primary" partitionName="1"/>
  </container>
```

```
  <container name="UNASSIGNED" zoneName="_ibm_SYSTEM"
  hostName="UNASSIGNED" serverName="UNNAMED">
    <shard type="SynchronousReplica" partitionName="0"/>
    <shard type="AsynchronousReplica" partitionName="0"/>
  </container>
</objectGrid>
```

Managed Beans (MBeans) für die Verwaltung der Umgebung verwenden

Sie können verschiedene Typen von JMX-Beans (Java Management Extensions) verwenden, um Implementierungen zu verwalten und zu überwachen. Jede MBean bezieht sich auf eine bestimmte Entität, z. B. eine Map, eXtreme Scale, einen Server, eine Replikationsgruppe oder ein Replikationsgruppen-Member.

JMX-MBean-Schnittstellen und WebSphere eXtreme Scale

Jede MBean hat get-Methoden, die Attributwerte darstellen. Diese get-Methoden können nicht direkt über Ihr Programm aufgerufen werden. Die JMX-Spezifikation behandelt Attribute anders als Operationen. Sie können Attribute über die JMX-Konsole eines anderen Anbieters anzeigen, und Sie können Operationen in Ihrem Programm oder über eine JMX-Konsole eines anderen Anbieters durchführen.

Zugehörige Konzepte

„Übersicht über Statistiken" auf Seite 263
Statistiken in WebSphere eXtreme Scale werden über eine interne Statistikstruktur erstellt. Die StatsAccessor-API, die PMI-Module (Performance Monitoring Infrastructure) und die MBean-API werden aus der internen Struktur erstellt.

„Tool eines anderen Anbieters" auf Seite 290
WebSphere eXtreme Scale kann mit verschiedenen gängigen Lösungen für die Unternehmensüberwachung überwacht werden. Es werden Plug-in-Agenten für IBM Tivoli Monitoring and Hyperic HQ bereitgestellt, die WebSphere eXtreme Scale mit Hilfe öffentlich zugänglicher Management-Beans überwachen. CA Wily Introscope verwendet die Java-Methodeninstrumentierung, um Statistiken zu erfassen.

Zugehörige Tasks

Kapitel 8, „Implementierungsumgebung überwachen", auf Seite 263
Sie können Anwendungsprogrammierschnittstellen (API, Application Programming Interface), MBeans, Protokolle und Dienstprogramme verwenden, um die Leistung Ihrer Anwendungsumgebung zu überwachen.

„Überwachung mit der API ″Statistics″" auf Seite 267
Die API ″Statistics″ ist die direkte Schnittstelle zur internen Statistikstruktur. Statistiken sind standardmäßig inaktiviert, können aber über die Definition einer Schnittstelle ″StatsSpec″ aktiviert werden. Eine Schnittstelle ″StatsSpec″ definiert, wie WebSphere eXtreme Scale Statistiken überwachen soll.

„Leistung mit WebSphere Application Server PMI überwachen" auf Seite 271
WebSphere eXtreme Scale unterstützt Performance Monitoring Infrastructure (PMI), wenn Sie mit einem Anwendungsserver von WebSphere Application Server oder WebSphere Extended Deployment arbeiten. PMI erfasst Leistungsdaten zu Laufzeitanwendungen und stellt Schnittstellen bereit, über die externe Anwendungen für die Überwachung von Leistungsdaten unterstützt werden. Sie können die Administrationskonsole oder das Tool ″wsadmin″ verwenden, um auf Überwachungsdaten zuzugreifen.

Übersicht über die Verwendung von Managed Beans

Sie können Managed Beans (MBeans) verwenden, um Statistiken in Ihrer Umgebung zu verfolgen.

Vorbereitungen

Für die aufzuzeichnenden Attribute müssen Sie die Statistiken aktivieren. Sie können Statistiken mit den folgenden Methoden aktivieren:

- **Über die Servereigenschaftendatei:**

  Sie können Statistiken in der Servereigenschaftendatei mit einem Schlüssel/Wert-Eintrag im Format `statsSpec=<StatsSpec>` aktivieren. Es folgenden verschiedene Beispiele für mögliche Einstellungen:

  – Zum Aktivieren aller Statistiken verwenden Sie `statsSpec=all=enabled` oder `statisticsSpec=all=enabled`.

  – Wenn Sie nur ObjectGrid-Statistiken aktivieren möchten, verwenden Sie `statsSpec=og.all=enabled`. Eine Beschreibung aller möglichen Statistikspezifikationen finden Sie in der API-Dokumentation zur API ″StatsSpec″.

  Weitere Informationen zur Servereigenschaftendatei finden Sie im Abschnitt „Servereigenschaftendatei" auf Seite 197.

- **Programmgesteuert:**

Sie können Statistiken auch programmgesteuert mit der Schnittstelle ″StatsAccessor″ aktivieren, die über die Klasse ″ StatsAccessorFactory″ abgerufen wird. Verwenden Sie diese Schnittstelle in einer Clientumgebung, oder wenn Sie eine Instanz von eXtreme Scale überwachen müssen, die im aktuellen Prozess ausgeführt wird.

Beispiel

Ein Beispiel für die Verwendung von Managed Beans finden Sie im Abschnitt „Musterdienstprogramm ″xsAdmin″ verwenden”.

Musterdienstprogramm ″xsAdmin″ verwenden

Mit dem Musterdienstprogramm ″xsAdmin″ können Sie Textinformationen zu Ihrer WebSphere eXtreme Scale-Topologie formatieren und anzeigen. Das Musterdienstprogramm stellt eine Methode für die Syntaxanalyse und die Erkennung aktuelle Implementierungsdaten bereit und kann als Grundlage für das Schreiben angepasster Dienstprogramme verwendet werden.

Vorbereitungen

Sie müssen WebSphere eXtreme Scale installiert haben.

Warum und wann dieser Vorgang ausgeführt wird

Sie können das Musterdienstprogramm ″xsAdmin″ verwenden, um Feedback zum aktuellen Layout und spezifischen Status des Grids, z. B. zum Map-Inhalt, bereitzustellen. In diesem Beispiel besteht das Layout des Grids in dieser Task aus einem einzigen Grid mit dem Namen *ObjectGridA* mit einer definierten Map mit dem Namen *MapA*, die zum MapSet mit dem Namen *MapSetA* gehört. Dieses Beispiel demonstriert, wie Sie alle aktiven Container in einem Grid anzeigen und gefilterte Metriken bezüglich der Map-Größe von *MapA* ausgeben. Zum Anzeigen aller möglichen Befehlsoptionen führen Sie das Dienstprogramm ″xsAdmin″ ohne Argumente oder mit der Option **-help** aus.

1. Setzen Sie über die Befehlszeile die Umgebungsvariable JAVA_HOME.
   - `export JAVA_HOME=javaHome`
   - `set JAVA_HOME=javaHome`
2. Navigieren Sie zum Verzeichnis ″bin″.
   `cd ObjectGrid-Stammverzeichnis/bin`
3. Starten Sie das Dienstprogramm ″xsAdmin″.
   - **Zum Anzeigen der Onlinehilfe führen Sie den folgenden Befehl aus:**
     `xsadmin.sh`

     `xsadmin.bat`

     Sehen Sie sich den Abschnitt mit den erforderlichen Argumenten in der Hilfenachricht an, weil Sie nur eine einzige der aufgelisteten Optionen übergeben dürfen, damit das Dienstprogramm funktioniert. Wenn Sie keine Option **-g** oder **-m** angeben, gibt das Dienstprogramm ″xsAdmin″ Informationen zu jedem Grid in der Topologie aus.
   - **Wenn Sie alle Onlinecontainer für ein Grid anzeigen möchten, führen Sie den folgenden Befehl aus:**

```
xsadmin.sh -g ObjectGridA -m MapSetA -containers
```

```
xsadmin.bat -g ObjectGridA -m MapSetA -containers
```

Alle Containerinformationen werden angezeigt. Es folgt ein Beispiel für die Ausgabe:

```
 This administrative utility is provided as a sample only and is not to be
considered a fully supported component of the WebSphere eXtreme Scale product
Connecting to Catalog service at localhost:1099
*** Show all online containers for grid - ObjectGridA & mapset - MapSetA
Host: 192.168.0.186
  Container: server1_C-0, Server:server1, Zone:DefaultZone
    P:0 Primary
  Num containers matching = 1
  Total known containers = 1
  Total known hosts = 1
```

- **Wenn Sie die Anzahl der Einträge in allen Maps für ein Grid anzeigen möchten, führen Sie den folgenden Befehl aus:**

```
xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA
```

```
xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA
```

Die Größe der angegebenen Map wird angezeigt. Es folgt ein Beispiel für die Ausgabe:

```
This administrative utility is provided as a sample only and is not to be
considered a fully supported component of the WebSphere eXtreme Scale product

Connecting to Catalog service at localhost:1099

****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


*** Listing Maps for server1 ***
 Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
Server Total: 0
```

- **Wenn Sie den JMX-Port für den Katalogservice angeben möchten, führen Sie den folgenden Befehl aus:** Das Musterdienstprogramm "xsAdmin" stellt eine Verbindung zum MBean-Server her, der in einem Katalogserver ausgeführt wird. Ein Katalogserver kann in einem eigenständigen Prozess, einem Prozess von WebSphere Application Server oder integriert in einem angepassten Anwendungsprozess ausgeführt werden. Verwenden Sie die Option **-ch**, um den Hostnamen des Katalogservice anzugeben, und die Option **-p**, um den Port des Katalogservice anzugeben.

```
xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  -ch CatalogMachine -p 6645
```

```
xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  -ch CatalogMachine -p 6645
```

Die Größe der angegebenen Map wird angezeigt. Es folgt ein Beispiel für die Ausgabe:

```
This administrative utility is provided as a sample only and is not to be
considered a fully supported component of the WebSphere eXtreme Scale product

Connecting to Catalog service at CatalogMachine:6645
```

```
*****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


*** Listing Maps for server1 ***
 Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
Server Total: 0
```

- **Wenn Sie eine Verbindung zu einem Katalogservice in einem Prozess von WebSphere Application Server herzustellen, führen Sie die folgenden Schritte aus:**

  Die Option **-dmgr** ist erforderlilch, wenn eine Verbindung zu einem Katalogservice in einem Prozess oder Prozesscluster von WebSphere Application Server hergestellt werden soll. Verwenden Sie die Option **-ch**, um den Hostnamen anzugeben, falls dieser nicht `localhost` ist, und die Option **-p**, um den Bootstrap-Port des Katalogservice zu überschreiben, der den mit BOOTSTRAP_ADDRESS angegebenen Prozess verwendet. Die Option **-p** ist nur erforderlich, wenn BOOTSTRAP_ADDRESS nicht auf den Standardwert von 9809 gesetzt ist.

  Anmerkung: Die eigenständige Version von WebSphere eXtreme Scale kann nicht verwendet werden, um eine Verbindung zu einem Katalogservice in einem Prozess von WebSphere Application Server herzustellen. Verwenden Sie das Script "xsAdmin" im Verzeichnis *WAS-Stammverzeichnis*`/bin`, das verfügbar ist, wenn Sie WebSphere eXtreme Scale in WebSphere Application Server oder WebSphere Application Server Network Deployment installieren.

  a. Navigieren Sie zum Verzeichnis "bin" von WebSphere Application Server.
     ```
     cd WAS-Stammverzeichnis/bin
     ```
  b. Starten Sie das Dienstprogramm "xsAdmin" mit dem folgenden Befehl:
     ```
     xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr


     xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
     ```
  c. Die Größe der angegebenen Map wird angezeigt.

```
This administrative utility is provided as a sample only and is not to be
considered a fully supported component of the WebSphere eXtreme Scale product

Connecting to Catalog service at localhost:9809

****Displaying Results for Grid - ObjectGridA, MapSet - MapSetA*****


*** Listing Maps for server1 ***
 Map Name: MapA  Partition #: 0  Map Size: 0  Shard Type: Primary
Server Total: 0
```

Zugehörige Konzepte

„Übersicht über Statistiken" auf Seite 263
Statistiken in WebSphere eXtreme Scale werden über eine interne Statistikstruktur erstellt. Die StatsAccessor-API, die PMI-Module (Performance Monitoring Infrastructure) und die MBean-API werden aus der internen Struktur erstellt.

Zugehörige Verweise

„PMI-Module" auf Seite 276
Sie können die Leistung Ihrer Anwendungen mit den PMI-Modulen (Performance Monitoring Infrastructure) überwachen.

Tool eines anderen Anbieters

WebSphere eXtreme Scale kann mit verschiedenen gängigen Lösungen für die Unternehmensüberwachung überwacht werden. Es werden Plug-in-Agenten für IBM Tivoli Monitoring and Hyperic HQ bereitgestellt, die WebSphere eXtreme Scale mit Hilfe öffentlich zugänglicher Management-Beans überwachen. CA Wily Introscope verwendet die Java-Methodeninstrumentierung, um Statistiken zu erfassen.

Zugehörige Tasks

Kapitel 8, „Implementierungsumgebung überwachen", auf Seite 263
Sie können Anwendungsprogrammierschnittstellen (API, Application Programming Interface), MBeans, Protokolle und Dienstprogramme verwenden, um die Leistung Ihrer Anwendungsumgebung zu überwachen.

„Überwachung mit der API ″Statistics″" auf Seite 267
Die API ″Statistics″ ist die direkte Schnittstelle zur internen Statistikstruktur. Statistiken sind standardmäßig inaktiviert, können aber über die Definition einer Schnittstelle ″StatsSpec″ aktiviert werden. Eine Schnittstelle ″StatsSpec″ definiert, wie WebSphere eXtreme Scale Statistiken überwachen soll.

„Leistung mit WebSphere Application Server PMI überwachen" auf Seite 271
WebSphere eXtreme Scale unterstützt Performance Monitoring Infrastructure (PMI), wenn Sie mit einem Anwendungsserver von WebSphere Application Server oder WebSphere Extended Deployment arbeiten. PMI erfasst Leistungsdaten zu Laufzeitanwendungen und stellt Schnittstellen bereit, über die externe Anwendungen für die Überwachung von Leistungsdaten unterstützt werden. Sie können die Administrationskonsole oder das Tool ″wsadmin″ verwenden, um auf Überwachungsdaten zuzugreifen.

„eXtreme Scale mit Hyperic HQ überwachen" auf Seite 301
Hyperic HQ ist eine Überwachungslösung eines anderen Anbieters, die kostenlos als Open-Source-Lösung oder als Unternehmensprodukt verfügbar ist. WebSphere eXtreme Scale enthält ein Plug-in, mit dem Hyperic-HQ-Agenten eXtreme-Scale-Containerserver erkennen und Statistiken mit Hilfe von eXtreme-Scale-Management-Beans berichten und zusammenfassen können. Sie können Hyperic HQ verwenden, um eigenständige Implementierungen von eXtreme Scale zu überwachen.

„Überwachung mit IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale"
IBM Tivoli Enterprise Monitoring Agent ist eine mit vielen Funktionen ausgestattete Überwachungslösung, die Sie verwenden können, um Datenbanken, Betriebssysteme und Server in verteilten Umgebungen und in Hostumgebungen zu überwachen. WebSphere eXtreme Scale enthält einen angepassten Agenten, den Sie verwenden können, um eXtreme-Scale-Management-Beans selbst zu überwachen. Diese Lösung funktioniert effizient in eigenständigen Implementierungen von eXtreme Scale und in Implementierungen von eXtreme Scale mit WebSphere Application Server.

Zugehörige Verweise

„Managed Beans (MBeans) für die Verwaltung der Umgebung verwenden" auf Seite 285
Sie können verschiedene Typen von JMX-Beans (Java Management Extensions) verwenden, um Implementierungen zu verwalten und zu überwachen. Jede MBean bezieht sich auf eine bestimmte Entität, z. B. eine Map, eXtreme Scale, einen Server, eine Replikationsgruppe oder ein Replikationsgruppen-Member.

Überwachung mit IBM Tivoli Enterprise Monitoring Agent for WebSphere eXtreme Scale

IBM Tivoli Enterprise Monitoring Agent ist eine mit vielen Funktionen ausgestattete Überwachungslösung, die Sie verwenden können, um Datenbanken, Betriebssysteme und Server in verteilten Umgebungen und in Hostumgebungen zu überwachen. WebSphere eXtreme Scale enthält einen angepassten Agenten, den Sie verwenden können, um eXtreme-Scale-Management-Beans selbst zu überwachen. Diese Lösung funktioniert effizient in eigenständigen Implementierungen von eXtreme Scale und in Implementierungen von eXtreme Scale mit WebSphere Application Server.

Vorbereitungen

- Installieren Sie WebSphere eXtreme Scale Version 7.0.0 oder höher.
- Installieren Sie IBM Tivoli Monitoring Version 6.2.1 mit Fixpack 2 oder höher.
- Installieren Sie den Tivoli-Betriebssystemagenten auf jedem Server oder Host, auf dem eXtreme-Scale-Server ausgeführt werden.
- Installieren Sie den WebSphere eXtreme Scale-Agenten, den Sie kostenlos von der Website IBM Open Process Automation Library (OPAL) herunterladen können.

Führen Sie die folgenden Schritte aus, um Tivoli Monitoring Agent zu installieren und zu konfigurieren:

1. Installieren Sie Tivoli Monitoring Agent for WebSphere eXtreme Scale.

   Laden Sie das Tivoli-Installations-Image herunter, und extrahieren Sie die Dateien in einem temporären Verzeichnis.
2. Installieren Sie die Unterstützungsdateien für eXtreme-Scale-Anwendungen.

   Installieren Sie die eXtreme-Scale-Anwendungsunterstützung in jeder der folgenden Implementierungen:
   - Tivoli Enterprise Portal Server (TEPS)
   - Enterprise Desktop Client (TEPD)
   - Tivoli Enterprise Monitoring Server (TEMS)

   a. Starten Sie in dem erstellten temporären Verzeichnis ein neues Befehlsfenster, und führen Sie die entsprechende ausführbare Datei für Ihre Plattform aus. Das Installationsscript erkennt den Typ Ihrer Tivoli-Implementierung (TEMS, TEPD oder TEPS) automatisch. Sie können jeden beliebigen Typ auf einem einzelnen oder auf mehreren Hosts installieren, und alle drei Implementierungstypen erfordern die Installation der Unterstützungsdateien für die eXtreme-Scale-Agentenanwendung.

   b. Vergewissern Sie sich, dass die ausgewählten Optionen für die implementierten Tivoli-Komponenten im Fenster des Installationsprogramms korrekt sind. Klicken Sie auf **Next**.

   c. Geben Sie auf Anforderung Ihren Hostnamen und Ihre Verwaltungsberechtigungsnachweise an. Klicken Sie auf **Next**.

   d. Wählen Sie **Monitoring Agent for WebSphere eXtreme Scale** aus. Klicken Sie auf **Next**.

   e. Sie werden über die auszuführenden Installationsaktionen benachrichtigt. Klicken Sie auf **Next**. Der Fortschritt der Installation wird bis zum Ende hin angezeigt.

   Nach der Ausführung dieser Prozedur sind alle erforderlichen Anwendungsunterstützungsdateien für den eXtreme-Scale-Agenten installiert.
3. Installieren Sie den Agenten auf jedem eXtreme-Scale-Knoten.

   Sie installieren einen Tivoli-Betriebssystemagenten auf jedem Computer. Sie müssen diesen Agenten nicht konfigurieren oder starten. Verwenden Sie dasselbe Installations-Image, das Sie bereits im vorherigen Schritt verwendet haben, um die plattformspezifische ausführbare Datei auszuführen.

   Als Richtlinie können Sie verwenden, dass nur ein einziger Agent pro Host installiert werden muss. Jeder Agent kann mehrere Instanzen von eXtreme-Scale-Servern unterstützen. Um die beste Leistung zu erzielen, verwenden Sie eine Agenteninstanz für ungefähr 50 eXtreme-Scale-Server.

a. Klicken Sie in der Eingangsanzeige des Installationsassistenten auf **Next**, um die Anzeige zu öffnen, in der Sie Informationen zum Installationspfad angeben.
b. Geben Sie im Feld **Tivoli Monitoring installation directory** einen Wert ein, oder navigieren Sie zum Verzeichnis `C:\IBM\ITM` (oder `/opt/IBM/ITM`). Vergewissern Sie sich, dass der im Feld **Location for installable media** angezeigte Wert korrekt ist, und klicken Sie auf **Next**.
c. Wählen Sie die Komponenten aus, die Sie hinzufügen möchten, z. B. **Perform a local install of the solution**, und klicken Sie auf **Next**.
d. Wählen Sie die Anwendungen aus, für die Sie die Unterstützung hinzufügen möchten, z. B. **Monitoring Agent for WebSphere eXtreme Scale**, und klicken Sie auf **Next**.
e. Der Fortschritt der Installation wird angezeigt, bis alle Anwendungsunterstützungsdateien erfolgreich hinzugefügt wurden.

Anmerkung: Wiederholen Sie diese Schritte auf jedem eXtreme-Scale-Knoten. Sie können auch eine unbeaufsichtigte Installation durchführen. Weitere Informationen zur unbeaufsichtigten Installation finden Sie im Information Center von IBM Tivoli Monitoring.

4. Konfigurieren Sie den WebSphere eXtreme Scale-Agenten.

   Jeder installierte Agent muss für die Überwachung eines Katalogservers und oder eXtreme-Scale-Servers konfiguriert werden.

   Die Schritte zum Konfigurieren von Windows- und UNIX-Plattformen sind verschieden. Die Konfiguration für die Windows-Plattform erfolgt über die Benutzerschnittstelle **Manage Tivoli Monitoring Services**. Die Konfiguration für UNIX-Plattformen ist befehlszeilenbasiert.

   Verwenden Sie die folgenden Schritte, um den Agenten anfänglich unter Windows zu konfigurieren.

   a. Klicken Sie im Fenster **Manage Tivoli Enterprise Monitoring Services** auf **Start → All Programs → IBM Tivoli Monitoring → Manage Tivoli Monitoring Services**.
   b. Klicken Sie mit der rechten Maustaste auf **Monitoring Agent for WebSphere eXtreme Scale**, und wählen Sie die Option **Configure using default** aus, woraufhin ein Fenster erscheint, in dem Sie eine eindeutige Instanz des Agenten erstellen können.
   c. Wählen Sie einen eindeutigen Namen aus, z. B. `instance1`, und klicken Sie auf **Next**.

   • Wenn Sie eigenständige eXtreme-Scale-Server überwachen möchten, führen Sie die folgenden Schritte aus:

     a. Aktualisieren Sie die Java-Parameter, und stellen Sie sicher, dass der Wert für **Java Home** korrekt ist. JVM-Argumente können leer bleiben. Klicken Sie auf **Next**.
     b. Wählen Sie für **MBean server connection type** einen Typ aus. Verwenden Sie `JSR-160-Complaint Server` für eigenständige eXtreme-Scale-Server. Klicken Sie auf **Next**.
     c. Wenn die Sicherheit aktiviert ist, aktualisieren Sie die Werte in den Feldern **User ID** und **Password**. Übernehmen Sie den Wert im Feld **JMX service URL**. Sie überschreiben diesen Wert später. Übernehmen Sie den Wert im Feld **JMX Class Path Information**. Klicken Sie auf **Next**.

     Zum Konfigurieren der Server für den Agenten unter Windows führen Sie die folgenden Schritte aus:

a. Konfigurieren Sie Unterknoteninstanzen von eXtreme-Scale-Servern im Teilfenster **WebSphere eXtreme Scale Grid Servers**. Wenn keine Containerserver auf Ihrem Computer vorhanden sind, klicken Sie auf **Next**, um das Teilfenster für den Katalogservice aufzurufen.

b. Wenn mehrere eXtreme-Scale-Containerserver auf Ihrem Computer vorhanden sind, konfigurieren Sie den Agenten so, dass jeder einzelne Server überwacht wird.

c. Sie können so viele eXtreme-Scale-Server hinzufügen, wie Sie benötigen, sofern ihre Namen und Ports eindeutig sind. Klicken Sie dazu auf **New**. (Wenn ein eXtreme-Scale-Server gestartet wird, muss ein Wert für den JMX-Port angegeben werden.)

d. Nach der Konfiguration der Containerserver klicken Sie auf **Next**. Daraufhin wird das Teilfenster **WebSphere eXtreme Scale Catalog Servers** angezeigt.

e. Wenn Sie keine Katalogserver haben, klicken Sie auf **OK**. Wenn Sie Katalogserver haben, fügen Sie eine neue Konfiguration für jeden Server hinzu, so wie Sie es für die Containerserver getan haben. Wählen Sie auch hier einen eindeutigen Namen aus, vorzugsweise denselben Namen, den Sie auch beim Starten des Katalogservers verwendet haben. Klicken Sie auf **OK**, um die Konfiguration zu beenden.

- Wenn Sie Server für den Agenten in eXtreme-Scale-Servern überwachen möchten, die in einen Prozess von WebSphere Application Server integriert sind, führen Sie die folgenden Schritte aus:

  a. Aktualisieren Sie die Java-Parameter, und stellen Sie sicher, dass der Wert für **Java Home** korrekt ist. JVM-Argumente können leer bleiben. Klicken Sie auf **Next**.

  b. Wählen Sie im Feld **MBean server connection type** einen Typ aus. Wählen Sie die Version von WebSphere Application Server für Ihre Umgebung aus. Klicken Sie auf **Next**.

  c. Stellen Sie sicher, dass die Informationen zu WebSphere Application Server in dieser Anzeige korrekt sind. Klicken Sie auf **Next**.

  d. Fügen Sie nur eine einzige Unterknotendefinition hinzu. Legen Sie einen Namen für die Unterknotendefinition fest, aber aktualisieren Sie die Portdefinition nicht. In einer Umgebung mit WebSphere Application Server können Daten von allen Anwendungsserverprozessen erfasst werden, die vom Node Agent verwaltet werden, der auf dem Computer ausgeführt wird. Klicken Sie auf **Next**.

  e. Wenn keine Katalogserver in der Umgebung vorhanden sind, klicken Sie auf **OK**. Wenn Sie Katalogserver haben, fügen Sie eine neue Konfiguration für jeden Katalogserver hinzu, ähnlich wie bei den Containerservern. Wählen Sie einen eindeutigen Namen für den Katalogservice aus, vorzugsweise den Namen, den Sie auch beim Starten des Katalogservice verwendet haben. Klicken Sie auf **OK**, um die Konfiguration zu beenden.

Anmerkung: Die Containerserver müssen nicht zusammen mit dem Katalogservice in einem Prozess ausgeführt werden.

Jetzt sind der Agent und die Server konfiguriert und betriebsbereit. Klicken Sie im nächsten Fenster mit der rechten Maustaste auf `instance1`, um den Agenten zu starten.

Zum Konfigurieren des Agenten auf der UNIX-Plattform über die Befehlszeile, führen Sie die folgenden Schritte aus:

Es folgt ein Beispiel für eigenständige Server, die einen JSR160-konformen Verbindungstyp verwenden. Das Beispiel zeigt drei eXtreme-Scale-Container

auf dem einzelnen Host (rhea00b02), und die JMX-Listener-Adressen sind 15000,15001 und 15002. Es gibt keine Katalogserver.

Die Ausgabe des Konfigurationsdienstprogramms wird in *Kursivschrift mit fester Breite* angezeigt und die Benutzeraktion in **Fettschrift mit fester Breite**. (Wenn keine Benutzeraktion erforderlich ist, wird der Standardwert durch Drücken der Eingabetaste ausgewählt.)

```
rhea00b02 # ./itmcmd config -A xt
Agent configuration started...
Enter instance name (default is: ): inst1
Edit "Monitoring Agent for WebSphere eXtreme Scale" settings? [ 1=Yes, 2=No ] (default is: 1):
Edit 'Java' settings? [ 1=Yes, 2=No ] (default is: 1):
Java home (default is: C:\Program Files\IBM\Java50): /opt/OG61/java
Java trace level [ 1=Error, 2=Warning, 3=Information, 4=Minimum Debug, 5=Medium Debug, 6=Maximum Debug, 7=All ] (default is: 1):
JVM arguments (default is: ):
Edit 'Connection' settings? [ 1=Yes, 2=No ] (default is: 1):
MBean server connection type [ 1=JSR-160-Compliant Server, 2=WebSphere Application Server version 6.0,
3=WebSphere Application Server version 6.1, 4=WebSphere Application Server version 7.0 ] (default is: 1): 1
Edit 'JSR-160-Compliant Server' settings? [ 1=Yes, 2=No ] (default is: 1):
JMX user ID (default is: ):
Enter JMX password (default is: ):
Re-type : JMX password (default is: ):
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:port/objectgrid/MBeanServer):
----------------------------------------
JMX Class Path Information
JMX base paths (default is: ):
JMX class path (default is: ):
JMX JAR directories (default is: ):
Edit 'WebSphere eXtreme Scale Catalog Service' settings? [ 1=Yes, 2=No ] (default is: 1): 2
Edit 'WebSphere eXtreme Scale Grid Servers' settings? [ 1=Yes, 2=No ] (default is: 1): 1
No 'WebSphere eXtreme Scale Grid Servers' settings available?
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 1
WebSphere eXtreme Scale Grid Servers (default is: ): rhea00b02_c0
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer):
service:jmx:rmi:///jndi/rmi://localhost:15000/objectgrid/MBeanServer

'WebSphere eXtreme Scale Grid Servers' settings:  WebSphere eXtreme Scale Grid Servers=ogx
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 1
WebSphere eXtreme Scale Grid Servers (default is: ): rhea00b02_c1
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer):
service:jmx:rmi:///jndi/rmi://localhost:15001/objectgrid/MBeanServer

'WebSphere eXtreme Scale Grid Servers' settings:  WebSphere eXtreme Scale Grid Servers= rhea00b02_c1
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 1
WebSphere eXtreme Scale Grid Servers (default is: ): rhea00b02_c2
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer):
service:jmx:rmi:///jndi/rmi://localhost:15002/objectgrid/MBeanServer

'WebSphere eXtreme Scale Grid Servers' settings:  WebSphere eXtreme Scale Grid Servers= rhea00b02_c2
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 5

Will this agent connect to a TEMS? [1=YES, 2=NO] (Default is: 1):
TEMS Host Name (Default is: rhea00b00):

Network Protocol [ip, sna, ip.pipe or ip.spipe] (Default is: ip.pipe):

     Now choose the next protocol number from one of these:
     - ip
     - sna
     - ip.spipe
     - 0 for none
Network Protocol 2 (Default is: 0):
IP.PIPE Port Number (Default is: 1918):
Enter name of KDC_PARTITION (Default is: null):

Configure connection for a secondary TEMS? [1=YES, 2=NO] (Default is: 2):
Enter Optional Primary Network Name or 0 for "none" (Default is: 0):
Agent configuration completed...
```

Der vorherige Beispielcode erstellt eine Agenteninstanz mit dem Namen ″inst1″ und aktualisiert die Einstellungen für das Java-Ausgangsverzeichnis. Die eXtreme-Scale-Containerserver sind konfiguriert, aber der Katalogserver ist nicht konfiguriert.

Anmerkung: Die vorherige Prozedur erstellt eine Textdatei im folgenden Format im Verzeichnis `<ITM-Installation>/config/<Host>_xt_<Instanzname>.cfg`.

Beispiel: `rhea00b02_xt_inst1.cfg`

Es empfiehlt sich, diese Datei mit einem Texteditor Ihrer Wahl zu editieren. Ein Beispiel für den Inhalt einer solchen Datei folgt:

```
INSTANCE=inst2 [        SECTION=KQZ_JAVA [ { JAVA_HOME=/opt/OG61/java }  { JAVA_TRACE_LEVEL=ERROR }  ]
SECTION=KQZ_JMX_CONNECTION_SECTION [ { KQZ_JMX_CONNECTION_PROPERTY=KQZ_JMX_JSR160_JSR160 }  ]
SECTION=KQZ_JMX_JSR160_JSR160 [ { KQZ_JMX_JSR160_JSR160_CLASS_PATH_TITLE= }
{ KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:rmi:///jndi/rmi://localho
st:port/objectgrid/MBeanServer }  { KQZ_JMX_JSR160_JSR160_CLASS_PATH_SEPARATOR= }  ]
SECTION=OGS:rhea00b02_c1 [ { KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:
rmi:///jndi/rmi://localhost:15001/objectgrid/MBeanServer }  ]
SECTION=OGS:rhea00b02_c0 [ { KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:
rmi:///jndi/rmi://localhost:15002/objectgrid/MBeanServer }  ]
SECTION=OGS:rhea00b02_c2 [ { KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:
rmi:///jndi/rmi://localhost:15002/objectgrid/MBeanServer }  ]]
```

Im Folgenden sehen Sie ein Beispiel, das die Konfiguration einer Implementierung von WebSphere Application Server zeigt:

```
rhea00b02 # ./itmcmd config -A xt
Agent configuration started...
Enter instance name (default is: ): inst1
Edit "Monitoring Agent for WebSphere eXtreme Scale" settings? [ 1=Yes, 2=No ] (default is: 1): 1
Edit 'Java' settings? [ 1=Yes, 2=No ] (default is: 1): 1
Java home (default is: C:\Program Files\IBM\Java50): /opt/WAS61/java
Java trace level [ 1=Error, 2=Warning, 3=Information, 4=Minimum Debug, 5=Medium Debug, 6=Maximum Debug, 7=All ] (default is: 1):
JVM arguments (default is: ):
Edit 'Connection' settings? [ 1=Yes, 2=No ] (default is: 1):
MBean server connection type [ 1=JSR-160-Compliant Server, 2=WebSphere Application Server version 6.0,
3=WebSphere Application Server version 6.1, 4=WebSphere Application Server version 7.0 ] (default is: 1): 4
Edit 'WebSphere Application Server version 7.0' settings? [ 1=Yes, 2=No ] (default is: 1):WAS user ID (default is: ):
Enter WAS password (default is: ):
Re-type : WAS password (default is: ):
WAS host name (default is: localhost): rhea00b02
WAS port (default is: 2809):
WAS connector protocol [ 1=rmi, 2=soap ] (default is: 1):
WAS profile name (default is: ): default
----------------------------------------
WAS Class Path Information
WAS base paths (default is: C:\Program Files\IBM\WebSphere\AppServer;/opt/IBM/WebSphere/AppServer): /opt/WAS61
WAS class path (default is: runtimes/com.ibm.ws.admin.client_6.1.0.jar;runtimes/com.ibm.ws.ejb.thinclient_7.0.0.jar):
WAS JAR directories (default is: lib;plugins):
Edit 'WebSphere eXtreme Scale Grid Servers' settings? [ 1=Yes, 2=No ] (default is: 1):
No 'WebSphere eXtreme Scale Grid Servers' settings available?
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 1
WebSphere eXtreme Scale Grid Servers (default is: ): rhea00b02
JMX service URL (default is: service:jmx:rmi:///jndi/rmi://localhost:<port>/objectgrid/MBeanServer):

'WebSphere eXtreme Scale Grid Servers' settings:  WebSphere eXtreme Scale Grid Servers=rhea00b02
Edit 'WebSphere eXtreme Scale Grid Servers' settings, [1=Add, 2=Edit, 3=Del, 4=Next, 5=Exit] (default is: 4): 5
Edit 'WebSphere eXtreme Scale Catalog Service' settings? [ 1=Yes, 2=No ] (default is: 1): 2
Will this agent connect to a TEMS? [1=YES, 2=NO] (Default is: 1):
TEMS Host Name (Default is: rhea00b02):

Network Protocol [ip, sna, ip.pipe or ip.spipe] (Default is: ip.pipe):

     Now choose the next protocol number from one of these:
     - ip
     - sna
     - ip.spipe
     - 0 for none
Network Protocol 2 (Default is: 0):
IP.PIPE Port Number (Default is: 1918):
Enter name of KDC_PARTITION (Default is: null):

Configure connection for a secondary TEMS? [1=YES, 2=NO] (Default is: 2):
Enter Optional Primary Network Name or 0 for "none" (Default is: 0):
Agent configuration completed...
rhea00b02 #
```

Für Implementierungen von WebSphere Application Server müssen Sie nicht mehrere Unterknoten erstellen. Der eXtreme-Scale-Agent stellt die Verbindung zum Node Agent her, um alle Informationen von den Anwendungsservern zu erfassen, für die er zuständig ist.

`SECTION=CAT` bezeichnet eine Katalogservicezeile, wohingegen `SECTION=OGS` eine Konfigurationszeile für einen eXtreme-Scale-Server bezeichnet.

5. Konfigurieren Sie die eXtreme-Scale-Server.

   Wenn eXtreme-Scale-Containerserver ohne das Argument **-JMXServicePort** gestartet werden, wird einem MBean-Server ein dynamischer Port zugeordnet. Der Agent muss im Voraus wissen, mit welchem Port er zu kommunizieren hat. Der Agent funktioniert nicht mit dynamischen Ports.

   Wenn Sie die Server starten, müssen Sie das Argument `-JMXServicePort <Portnummer>` angeben, wenn Sie den eXtreme-Scale-Server mit dem Befehl

`startOgServer.sh | .bat` starten. Die Ausführung dieses Befehls gewährleistet, dass der JMX-Server im Prozess an einem statischen vordefinierten Port empfangsbereit ist.

Für die vorherigen Beispiele in einer UNIX-Installation müssen zwei eXtreme-Scale-Server mit definierten Ports gestartet werden:

a. ″-JMXServicePort″ ″15000″ (für rhea00b02_c0)
b. ″-JMXServicePort″ ″15001″ (für rhea00b02_c1)

a. Starten Sie den WebSphere eXtreme Scale-Agenten.

   Davon ausgehend, dass wie im vorherigen Beispiel die Instanz `inst1` erstellt wurde, setzen Sie die folgenden Befehle ab:

   1) `cd <ITM-Installation>/bin`
   2) `itmcmd agent –o inst1 start xt`

b. Stoppen Sie den WebSphere eXtreme Scale-Agenten.

   Davon ausgehend, dass wie im vorherigen Beispiel die Instanz `inst1` erstellt wurde, setzen Sie die folgenden Befehle ab:

   1) `cd <ITM-Installation>/bin`
   2) `itmcmd agent –o inst1 stop xt`

Ergebnisse

Nachdem alle Server konfiguriert und gestartet wurden, werden MBean-Daten in der Konsole von IBM Tivoli Portal angezeigt. In vordefinierten Arbeitsbereichen werden Graphen und Datenmetriken auf jeder Knotenebene angezeigt.

Die folgenden Arbeitsbereiche sind definiert: **WebSphere eXtreme Scale Grid Servers** - für alle überwachten Knoten

- eXtreme Scale Transactions View
- eXtreme Scale Primary Shard View
- eXtreme Scale Memory View
- eXtreme Scale ObjectMap View

Sie können auch eigene Arbeitsbereiche konfigurieren. Weitere Informationen finden Sie in den Informationen zum Anpassen von Arbeitsbereichen im Information Center von IBM Tivoli Monitoring.

Zugehörige Konzepte

„Tool eines anderen Anbieters” auf Seite 290
WebSphere eXtreme Scale kann mit verschiedenen gängigen Lösungen für die Unternehmensüberwachung überwacht werden. Es werden Plug-in-Agenten für IBM Tivoli Monitoring and Hyperic HQ bereitgestellt, die WebSphere eXtreme Scale mit Hilfe öffentlich zugänglicher Management-Beans überwachen. CA Wily Introscope verwendet die Java-Methodeninstrumentierung, um Statistiken zu erfassen.

„Übersicht über Statistiken” auf Seite 263
Statistiken in WebSphere eXtreme Scale werden über eine interne Statistikstruktur erstellt. Die StatsAccessor-API, die PMI-Module (Performance Monitoring Infrastructure) und die MBean-API werden aus der internen Struktur erstellt.

eXtreme-Scale-Anwendungen mit CA Wily Introscope überwachen

CA Wily Introscope ist ein Managementprodukt eines anderen Anbieters, das Sie verwenden können, um Leistungsprobleme in Unternehmensanwendungsumge-

bungen zu erkennen und zu diagnostizieren. eXtreme Scale enthält Einstellungen für die Konfiguration von CA Wily Introscope für die Introspektion ausgewählter Komponenten der Laufzeitumgebung von eXtreme Scale, um eXtreme-Scale-Anwendungen schnell anzeigen und validieren zu können. CA Wily Introscope funktioniert effizient für eigenständige Implementierungen und Implementierungen von WebSphere Application Server.

Übersicht

Wenn Sie eXtreme-Scale-Anwendungen mit CA Wily Introscope überwachen möchten, müssen Sie Einstellungen in den PBD-Dateien (ProbeBuilderDirective) festlegen, die Ihnen Zugriff auf die Überwachungsinformationen für eXtreme Scale geben.

Achtung: Die Instrumentierungspunkte für Introscope können sich mit jedem Fixpack oder Release ändern. Wenn Sie ein neues Fixpack oder Release installieren, suchen Sie in der Dokumentation nach Hinweisen zu Änderungen bezüglich der Instrumentierungspunkte.

Sie können PBD-Dateien (ProbeBuilderDirective) von CA Wily Introscope konfigurieren, um Ihre eXtreme-Scale-Anwendungen zu überwachen. CA Wily Introscope ist ein Anwendungsmanagementprodukt, mit dem Sie Leistungsprobleme in komplexen, Verbund- und Webanwendungsumgebungen proaktiv erkennen, sichten und diagnostizieren können.

Einstellungen in der PBD-Datei für die Überwachung des Katalogservice

Sie können eine oder mehrere der folgenden Einstellungen in Ihrer PBD-Datei verwenden, um den Katalogservice zu überwachen:

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  changeDefinedCompleted
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  viewChangeCompleted
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  viewAboutToChange
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeat
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatCluster
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatCurrentLeader
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatDeadServer
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatNewLeader
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatNewServer
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  PlacementServiceImpl
importRouteInfo BlamePointTracerDifferentMethods
  "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  PlacementServiceImpl heartbeat
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  PlacementServiceImpl joinPlacementGroup
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
  TraceOneMethodOfClass:
com.ibm.ws.objectgrid.catalog.placement.PlacementServiceImpl
  classifyServer
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
```

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  BalanceGridEventListener shardActivated
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  BalanceGridEventListener shardDeactivate
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
```

Klassen für die Überwachung des Katalogservice

HAControllerImpl
Die Klasse ″HAControllerImpl″ verarbeitet Lebenszyklus- und Feedback-Ereignisse für die Stammgruppe. Sie können diese Klasse überwachen, um einen Hinweis auf die Struktur und Änderungen der Stammgruppe zu erhalten.

ServerAgent
Die Klasse ″ServerAgent″ ist für die Kommunikation von Stammgruppenereignissen an den Katalogservice zuständig. Sie können die verschiedenen Aufrufe für den Austausch von Überwachungssignalen überwachen, um wichtige Ereignisse zu erkennen.

PlacementServiceImpl
Die Klasse ″PlacementServiceImpl″ koordiniert die Container. Sie können die Methoden in dieser Klasse verwenden, um das Beitreten von Servern zur Stammgruppe und Verteilungsereignisse zu überwachen.

BalanceGridEventListener
Die Klasse ″BalanceGridEventListener″ steuert die Leitung des Katalogs. Sie können diese Klasse überwachen, um einen Hinweis auf den Katalogservice zu erhalten, der momentan als leitender Server agiert.

Einstellungen in der PBD-Datei für die Überwachung der Container

Sie können eine oder mehrere der folgenden Einstellungen in Ihrer PBD-Datei verwenden, um die Container zu überwachen:

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.ShardImpl processMessage
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.plugins.
  CommittedLogSequenceListenerProxy applyCommitted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.plugins.
  CommittedLogSequenceListenerProxy sendApplyCommitted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.map.BaseMap evictMapEntries
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.checkpoint.
  CheckpointMapImpl$CheckpointIterator activateListener
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  changeDefinedCompleted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  viewChangeCompleted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  viewAboutToChange
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  batchProcess
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeat
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatCluster
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatCurrentLeader
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatDeadServer
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatNewLeader
```

```
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatNewServer
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
```

Klassen für die Überwachung der Container

ShardImpl
Die Klasse ″ShardImpl″ hat die Methode ″processMessage. Die Methode ″processMessage″ ist die Methode für Clientanforderungen. Mit dieser Methode können Sie serverseitige Antwortzeiten und Anforderungszähler abrufen. Indem Sie die Zähler für alle Server und die Auslastung des Heap-Speichers überwachen, können Sie feststellen, ob das Grid ausgeglichen ist.

CheckpointIterator
Die Klasse ″CheckpointIterator″ hat die Methode ″activateListener″, die primäre Shards in den Peer-Modus versetzt. Wenn die primären Shards in den Peer-Modus versetzt werden, ist das Replikat nach Abschluss der Methode auf demselben Stand wie das primäre Shard. Wenn ein Replikat über ein vollständiges primäres Shard neu generiert wird, kann diese Operation längere Zeit dauern. Das System ist erst dann vollständig wiederhergestellt, wenn diese Operation abgeschlossen ist. Sie können diese Klasse verwenden, um den Fortschritt der Operation zu überwachen.

CommittedLogSequenceListenerProxy
Die Klasse ″CommittedLogSequenceListenerProxy″ hat zwei Methoden, die von Interesse sind. Die Methode ″applyCommitted″ wird für jede Transaktion ausgeführt, und die Methode ″sendApplyCommitted″ wird ausgeführt, wenn das Replikat Informationen extrahiert. Das Verhältnis, in dem die beiden Methoden ausgeführt werden, kann Ihnen einen Hinweis darauf geben, inwieweit das Replikat in der Lage ist, mit dem primären Shard Schritt zu halten.

Einstellungen in der PBD-Datei für die Überwachung der Clients

Sie können eine oder mehrere der folgenden Einstellungen in Ihrer PBD-Datei verwenden, um die Clients zu überwachen:

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.client.ORBClientCoreMessageHandler
 sendMessage
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.corba.cluster.ClusterStore
  bootstrap
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.corba.cluster.ClusterStore
  epochChangeBootstrap
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.map.BaseMap evictMapEntries
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.cluster.orb.routing.
  SelectionServiceImpl routeFailed
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.cluster.orb.routing.
  SelectionServiceImpl routeFailed
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.SessionImpl getMap
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.ObjectGridImpl getSession
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TurnOn: ObjectMap
SetFlag: ObjectMap
IdentifyClassAs: com.ibm.ws.objectgrid.ObjectMapImpl ObjectMap
TraceComplexMethodsifFlagged: ObjectMap BlamePointTracerDifferentMethods
"OGclient|{classname}|{method}"
```

Klassen für die Überwachung der Clients

ORBClientCoreMessageHandler
Die Klasse ″ORBClientCoreMessageHandler″ ist für das Senden von

Anwendungsanforderungen an die Container zuständig. Sie können die Methode ″sendMessage″ überwachen, um Clientantwortzeiten und Anforderungsanzahl zu erhalten.

ClusterStore
Die Klasse ″ClusterStore″ enthält Routing-Informationen auf der Clientseite.

BaseMap
Die Klasse ″BaseMap″ hat die Methode ″evictMapEntries″, die aufgerufen wird, wenn das Bereinigungsprogramm Einträge aus der Map entfernen möchte.

SelectionServiceImpl
Die Klasse ″SelectionServiceImpl″ trifft Routing-Entscheidungen. Wenn der Client Failover-Entscheidungen trifft, können Sie diese Klasse verwenden, um die aus den Entscheidungen resultierenden Aktionen zu überwachen.

ObjectGridImpl
Die Klasse ″ObjectGridImpl″ hat die Methode ″getSession″, die Sie überwachen können, um die Anzahl der Anforderungen an diese Methode zu erhalten.

eXtreme Scale mit Hyperic HQ überwachen

Hyperic HQ ist eine Überwachungslösung eines anderen Anbieters, die kostenlos als Open-Source-Lösung oder als Unternehmensprodukt verfügbar ist. WebSphere eXtreme Scale enthält ein Plug-in, mit dem Hyperic-HQ-Agenten eXtreme-Scale-Containerserver erkennen und Statistiken mit Hilfe von eXtreme-Scale-Management-Beans berichten und zusammenfassen können. Sie können Hyperic HQ verwenden, um eigenständige Implementierungen von eXtreme Scale zu überwachen.

Vorbereitungen

- Die folgenden Anweisungen gelten für Hyperic Version 4.0. Wenn Sie eine neuere Version von Hyperic haben, schlagen Sie in der Hyperic-Dokumentation Informationen wie Pfadnamen und Informationen zum Starten von Agenten und Servern nach.
- Laden Sie Hyperic-Server- und -Agenteninstallationen herunter. Eine Serverinstallation muss aktiv sein. Um alle eXtreme-Scale-Server zu erkennen, muss ein Hyperic-Agent auf jeder Maschine ausgeführt werden, auf dem ein eXtreme-Scale-Server ausgeführt wird. Downloadinformationen und Dokumentationsunterstützung finden Sie auf der Website von Hyperic.
- Sie müssen Zugriff auf die Dateien `objectgrid-plugin.xml` und `hqplugin.jar` haben. Diese Dateien befinden sich im Verzeichnis `ObjectGrid-Stammverzeichnis/hyperic/etc`.

Warum und wann dieser Vorgang ausgeführt wird

Durch die Integration der Überwachungssoftware Hyperic HQ in eXtreme Scale können Sie Metriken zur Leistung Ihrer Umgebung grafisch überwachen und anzeigen. Sie konfigurieren diese Integration über eine Plug-in-Implementierung in jedem Agenten.

1. Starten Sie Ihre eXtreme-Scale-Server. Das Hyperic-Plug-in prüft die lokalen Prozesse, um eine Verbindung zu den Java Virtual Machines herzustellen, in denen eXtreme Scale ausgeführt wird. Für eine ordnungsgemäße Verbindungsherstellung zu den Java Virtual Machines muss jeder Server mit der Option

-jmxServicePort gestartet werden. Außerdem müssen Sie die Statistiken in der Servereigenschaftendatei aktivieren. Katalogserver werden von diesem Filter nicht erkannt.

- Informationen zum Starten von Servern mit der Option **-jmxServicePort** finden Sie im Abschnitt „Script ″startOgServer″" auf Seite 235.
- Informationen zum Aktivieren der Statistiken in der Servereigenschaftendatei finden Sie in den Informationen zur Eigenschaft **statsSpec** im Abschnitt „Servereigenschaftendatei" auf Seite 197.

2. Kopieren Sie die Datei `extremescale-plugin.xml` und die Datei `wxshyperic.jar` in die entsprechenden Server- und Agenten-Plug-in-Verzeichnisse in Ihrer Hyperic-Konfiguration. Zum Integrieren von Hyperic müssen die Agenten- und die Serverinstallationen Zugriff auf das Plug-in und die JAR-Dateien haben. Obwohl der Server Konfigurationen dynamisch wechseln kann, müssen Sie die Integration vor dem Starten von Agenten durchführen.
   a. Kopieren Sie die Datei `extremescale-plugin.xml` in das Serververzeichnis `plugin` an der folgenden Position:
      ```
      hyperic_home/server_home/hq-engine/server/default/deploy/hq.ear/hq-plugins
      ```
   b. Kopieren Sie die Datei `extremescale-plugin.xml` in das Agentenverzeichnis `plugin` an der folgenden Position:
      ```
      agent_home/bundles/gent-4.0.2-939/pdk/plugins
      ```
   c. Kopieren Sie die Datei `wshyperic.jar` in das Agentenverzeichnis `lib` an der folgenden Position:
      ```
      agent_home/bundles/gent-4.0.2-939/pdk/lib
      ```
3. Konfigurieren Sie den Agenten. Die Datei `agent.properties` dient als Konfigurationspunkt für die Agentenlaufzeitumgebung. Diese Datei ist im Verzeichnis *agent_home*`/conf` enthalten. Die folgenden Schlüssel sind optional, aber wichtig für das eXtreme-Scale-Plug-in:
   - `autoinventory.defaultScan.interval.millis=<`*Zeit_in_Millisekunden*`>`

     Legt das Intervall (in Millisekunden) fest, in dem die Erkennungsoperationen des Agenten ausgeführt werden.
   - `log4j.logger.org.hyperic.hq.plugin.extremescale.XSServerDetector=DEBUG`

     : Aktiviert ausführliche Debug-Anweisungen vom eXtreme-Scale-Plug-in.
   - `username=<`*Benutzername*`>`: Legt den JMX-Benutzernamen (Java Management Extensions) fest, wenn die Sicherheit aktiviert ist.
   - `password=<`*Kennwort*`>`: Legt das JMX-Kennwort fest, wenn die Sicherheit aktiviert ist.
   - `sslEnabled=<`*true*`|`*false*`>`: Teilt dem Plug-in mit, ob Secure Sockets Layer (SSL) verwendet werden soll. Der Standardwert ist `false`.
   - `trustPath=<`*Pfad*`>`: Legt den Sicherheitspfad für die SSL-Verbindung fest.
   - `trustType=<`*Typ*`>`: Legt den Sicherheitstyp für die SSL-Verbindung fest.
   - `trustPass=<`*Kennwort*`>`: Legt das Sicherheitskennwort für die SSL-Verbindung fest.
4. Starten Sie die Agentenerkennung. Die Hyperic-Agenten senden Erkennungs- und Metrikinformationen an den Server. Verwenden Sie den Server, um Datensichten anzupassen und logische Bestandsobjekt zu gruppieren, um hilfreiche Informationen zu erhalten. Wenn der Server verfügbar ist, müssen Sie das Startscript ausführen oder den Windows-Dienst für den Agenten starten:
   - `agent_home/bin/hq-agent.sh start`

- Starten Sie den Agenten mit dem Windows-Dienst.

Nachdem Sie den Agenten gestartet haben, werden die Server erkannt und Gruppen konfiguriert. Sie können sich an der Serverkonsole anmelden und die Ressourcen auswählen, die der Bestandsdatenbank für den Server hinzugefügt werden sollen. Die Serverkonsole ist standardmäßig über den folgenden URL erreichbar: `http://<Hostname_des_Servers>:7080/`

5. Überwachen Sie Server mit der Hyperic-Konsole. Nachdem die Server dem Bestandsmodell hinzugefügt wurden, sind ihre Services nicht mehr erforderlich.
   - **Dashboard-Sicht**: Wenn Sie die Ressourcenerkennungsereignisse anzeigen möchten, müssen Sie sich bei der Haupt-Dashboard-Sicht anmelden. Die Dashboard-Sicht ist eine generische Sicht, die als Message Center dient, das Sie anpassen können. Sie können Graphen oder Bestandsobjekte in dieses Haupt-Dashboard exportieren.
   - **Ressourcensicht:** Sie können das gesamte Bestandsmodell über diese Seite abfragen und anzeigen. Nachdem Sie die Server hinzugefügt haben, wird jeder eXtreme-Scale-Server ordnungsgemäß beschriftet im Abschnitt mit den Servern aufgelistet. Sie können auf die einzelnen Server klicken, um die Basismetriken anzuzeigen.
6. Zeigen Sie den gesamten Serverbestand auf der Seite mit der Ressourcensicht an. Auf dieser Seite können Sie mehrere ObjectGrid-Server auswählen und gruppieren. Nach der Gruppierung von Ressourcen können die gemeinsamen Metriken der jeweiligen Gruppe in Graphen dargestellt werden, um Überschneidungen und Unterschiede zwischen den Gruppen-Membern anzuzeigen. Um eine Überschneidung anzuzeigen, wählen Sie die Metrik in der Anzeige Ihrer Servergruppe aus. Die Metrik wird dann im Diagrammbereich angezeigt. Zum Anzeigen einer Überschneidung für alle Gruppen-Member klicken Sie auf den unterstrichenen Metriknamen. Sie können alle Diagramme, Knotensichten und Vergleichsüberschneidungen über das Menü **Tools** in das Haupt-Dashboard exportieren.

Zugehörige Konzepte

„Tool eines anderen Anbieters” auf Seite 290
WebSphere eXtreme Scale kann mit verschiedenen gängigen Lösungen für die Unternehmensüberwachung überwacht werden. Es werden Plug-in-Agenten für IBM Tivoli Monitoring and Hyperic HQ bereitgestellt, die WebSphere eXtreme Scale mit Hilfe öffentlich zugänglicher Management-Beans überwachen. CA Wily Introscope verwendet die Java-Methodeninstrumentierung, um Statistiken zu erfassen.

„Übersicht über Statistiken” auf Seite 263
Statistiken in WebSphere eXtreme Scale werden über eine interne Statistikstruktur erstellt. Die StatsAccessor-API, die PMI-Module (Performance Monitoring Infrastructure) und die MBean-API werden aus der internen Struktur erstellt.

Protokolle und Trace

Sie können Protokolle und Trace verwenden, um Ihre Umgebung zu überwachen und Fehler zu beheben. Protokolle werden je nach Konfiguration an unterschiedlichen Positionen gespeichert. Möglicherweise müssen Sie einen Trace für einen Server bereitstellen, wenn Sie mit der IBM Unterstützungsfunktion zusammenarbeiten.

Protokolle mit WebSphere Application Server

Weitere Informationen finden Sie im Information Center von WebSphere Application Server.

Protokolle mit WebSphere eXtreme Scale in einer eigenständigen Umgebung

Bei eigenständigen Katalog- und Containerservern legen Sie die Position der Protokolle und alle Trace-Spezifikationen fest. Die Katalogserverprotokolle werden dort gespeichert, wo Sie den Befehl zum Starten des Servers ausführen.

Protokollposition für Containerserver festlegen

Standardmäßig werden die Protokolle für einen Server in dem Verzeichnis gespeichert, in dem der Befehl zum Starten des Servers ausgeführt wird. Wenn Sie die Server im Verzeichnis *<Ausgangsverzeichnis_von_eXtreme_Scale>*/bin starten, werden die Protokoll- und Trace-Dateien in den Verzeichnissen logs/*<Servername>* des Verzeichnisses bin gespeichert. Wenn Sie eine andere Position für die Protokolle eines Containerservers festlegen möchten, erstellen Sie eine Eigenschaftendatei, z. B. server.properties, mit dem folgenden Inhalt:

```
workingDirectory=<Verzeichnis>
traceSpec=
systemStreamToFileEnabled=true
```

Die Eigenschaft "workingDirectory" ist das Stammverzeichnis für die Protokolle und die optionale Trace-Datei. WebSphere eXtreme Scale erstellt ein Verzeichnis mit dem Namen des Containerservers mit einer Datei SystemOut.log, einer Datei SystemErr.log und einer Trace-Datei, falls die Trace-Erstellung mit der Option "traceSpec" aktiviert wurde. Wenn eine Eigenschaftendatei während des Containerstarts verwendet werden soll, verwenden Sie die Option **-serverProps**, und geben Sie die Position der Servereigenschaftendatei an.

Weitere Informationen finden Sie in den Abschnitten „Eigenständige Server von WebSphere eXtreme Scale starten" auf Seite 228 und „Script "startOgServer"" auf Seite 235.

Häufige Informationsnachrichten, die in der Datei SystemOut.log aufgezeichnet werden, sind Bestätigungsnachrichten für den Start. Weitere Informationen zu bestimmten Nachrichten finden Sie im Abschnitt „Nachrichten" auf Seite 335.

Trace-Erstellung mit WebSphere Application Server

Weitere Informationen finden Sie im Information Center von WebSphere Application Server.

Trace-Erstellung für den Katalogservice

Sie können die Trace-Erstellung für einen Katalogservice festlegen, indem Sie während des Katalogservicestarts die Parameter **-traceSpec** und **-traceFile** verwenden. Beispiel:

```
startOgServer.sh catalogServer -traceSpec
ObjectGridPlacement=all=enabled -traceFile
/home/user1/logs/trace.log
```

Wenn Sie den Katalogservice im Verzeichnis *<Ausgangsverzeichnis_von_eXtreme_Scale>*/bin starten, werden die Protokolle und Trace-Dateien in einem Verzeichnis logs/*<Name_des_Katalogservice>* des Verzeichnisses bin gespeichert. Weitere Einzelheiten zum Starten eines Katalogservice finden Sie im Abschnitt „Katalogservice in einer eigenständigen Umgebung starten" auf Seite 229.

Trace für einen eigenständigen Containerserver erstellen

Sie können die Trace-Erstellung für eine Containerserver auf zwei Arten aktivieren. Sie können, wie im Abschnitt zu den Protokollen erläutert, eine Servereigenschaftendatei erstellen, oder Sie können die Trace-Erstellung beim Start über die Befehlszeile aktivieren. Zum Aktivieren der Trace-Erstellung für den Container über eine Servereigenschaftendatei aktualisieren Sie die Eigenschaft **traceSpec** mit der erforderlichen Trace-Spezifikation. Zum Aktivieren der Trace-Erstellung für den Container über Startparameter verwenden Sie die Parameter **-traceSpec** und **-traceFile**. Beispiel:

```
startOgServer.sh c0 -objectGridFile ../xml/myObjectGrid.xml
-deploymentPolicyFile ../xml/myDepPolicy.xml -catalogServiceEndpoints
server1.rchland.ibm.com:2809 -traceSpec
ObjectGridPlacement=all=enabled -traceFile /home/user1/logs/trace.log
```

Wenn Sie den Server im Verzeichnis *<Ausgangsverzeichnis_von_eXtreme_Scale>*/bin starten, werden die Protokoll- und Trace-Dateien in den Verzeichnissen logs/*<Servername>* des Verzeichnisses bin gespeichert. Weitere Einzelheiten zum Starten eines Containerprozesses finden Sie im Abschnitt „Containerprozesse starten" auf Seite 232.

Trace-Erstellung mit der Schnittstelle "ObjectGridManager"

Eine weitere Option ist die Definition der Trace-Erstellung zur Laufzeit über eine ObjectGridManager-Schnittstelle. Die Definition der Trace-Erstellung in einer ObjectGridManager-Schnittstelle kann verwendet werden, um einen Trace für einen eXtreme-Scale-Client zu erstellen, wenn dieser eine Verbindung zu einer eXtreme-Scale-Instanz herstellt und Transaktionen festschreibt. Wenn Sie die Trace-Erstellung in einer ObjectGridManager-Schnittstelle festlegen möchten, geben Sie eine Trace-Spezifikation und ein Trace-Protokoll an.

```
ObjectGridManager manager = ObjectGridManagerFactory.getObjectGridManager();
...
manager.setTraceEnabled(true);
manager.setTraceFileName("logs/myClient.log");
manager.setTraceSpecification("ObjectGridReplication=all=enabled");
```

Trace-Erstellung mit dem Dienstprogramm "xsadmin" aktivieren

Zum Aktivieren der Trace-Erstellung mit dem Dienstprogramm "xsadmin" verwenden Sie die Option **setTraceSpec**. Verwenden Sie das Dienstprogramm "xsadmin", um die Trace-Erstellung für eine eigenständige Umgebung zur Laufzeit und nicht zur Startzeit zu aktivieren. Sie können die Trace-Erstellung für alle Server und Katalogservices aktivieren, oder Sie können die Server nach dem ObjectGrid-Namen usw. filtern. Wenn Sie beispielsweise den ObjectGridReplication-Trace mit Zugriff auf den Katalogserviceserver aktivieren möchten, führen Sie den folgenden Befehl aus:

```
<Ausgangsverzeichnis_von_eXtreme_Scale>/bin>xsadmin.bat -setTraceSpec "ObjectGridReplication=all=enabled"
```

Sie können die Trace-Erstellung auch inaktivieren, indem Sie die Trace-Spezifikation auf *=all=disabled. setzen.

Weitere Informationen finden Sie im Abschnitt „Musterdienstprogramm "xsAdmin" verwenden" auf Seite 287.

FFDC-Verzeichnis und -Dateien

FFDC-Dateien sind als Debug-Hilfe für die IBM Unterstützungsfunktion bestimmt. Diese Dateien werden möglicherweise von der IBM Unterstützungsfunktion angefordert, wenn ein Problem auftritt.

Diese Dateien befinden sich in einem Verzeichnis mit dem Namen `ffdc`. Das Verzeichnis enthält Dateien wie die folgenden:

```
server2_exception.log
server2_20802080_07.03.05_10.52.18_0.txt
```

Zugehörige Konzepte

„Sichere eXtreme-Scale-Server starten und stoppen" auf Seite 327
Servers müssen für die Implementierungsumgebung häufig sicher sein, und dies erfordert eine spezielle Konfiguration.

Zugehörige Tasks

„Eigenständige Server von WebSphere eXtreme Scale starten" auf Seite 228
Wenn Sie eine eigenständige Konfiguration von WebSphere eXtreme Scale verwenden, setzt sich die Umgebung aus Katalogservern, Containerservern und eXtreme-Scale-Clientprozessen zusammen. Sie müssen diese Prozesse manuell konfigurieren und starten.

„Katalogservice in einer eigenständigen Umgebung starten" auf Seite 229
Sie müssen den Katalogservice manuell starten, wenn Sie eine verteilte Umgebung von WebSphere eXtreme Scale ohne WebSphere Application Server verwenden.

„Containerprozesse starten" auf Seite 232
Sie können eXtreme Scale über die Befehlszeile, über eine Implementierungstopologie oder über eine Datei `server.properties` starten.

Trace-Optionen

Sie können die Trace-Erstellung aktivieren, um der IBM Unterstützungsfunktion Informationen über Ihre Umgebung bereitzustellen.

Informationen zur Trace-Erstellung

Der WebSphere eXtreme Scale-Trace ist in mehrere Komponenten unterteilt. Ähnlich wie bei der Trace-Erstellung in WebSphere Application Server können Sie die zu verwendende Trace-Stufe angeben. Zu den gebräuchlichen Trace-Stufen gehören all, debug, entryExit und event.

Im Folgenden sehen Sie ein Beispiel für eine Trace-Zeichenfolge:

```
ObjectGridComponent=level=enabled
```

Sie können Trace-Zeichenfolgen verknüpfen. Verwenden Sie das Symbol `*` (Stern), um einen Platzhalterwert anzugeben, z. B. `ObjectGrid*=all=enabled`. Wenn Sie einen Trace für die IBM Unterstützungsfunktion bereitstellen müssen, ist eine bestimmte Trace-Zeichenfolge erforderlich. So kann beispielsweise die Trace-Zeichenfolge `ObjectGridReplication=debug=enabled` angefordert werden, wenn ein Problem mit der Replikation auftritt.

Trace-Spezifikation

ObjectGrid
Allgemeine Basiscachesteuerkomponente.

ObjectGridCatalogServer
Allgemeiner Katalogservice.

ObjectGridChannel
Statische Kommunikation in der Implementierungstopologie.

ObjectgridCORBA
Dynamische Kommunikation in der Implementierungstopologie.

ObjectGridDataGrid
Die API ″AgentManager″.

ObjectGridDynaCache
Der dynamische Cacheprovider von WebSphere eXtreme Scale.

ObjectGridEntityManager
Die API ″EntityManager″. Mit der Option ″Projector″ zu verwenden.

ObjectGridEvictors
Integrierte ObjectGrid-Bereinigungsprogramme.

ObjectGridJPA
JPA-Loader (Java Persistence API).

ObjectGridJPACache
JPA-Cache-Plug-ins.

ObjectGridLocking
Sperrenmanager für ObjectGrid-Cacheeinträge.

ObjectGridMBean
Management-Beans.

ObjectGridPlacement
Catalogserverservice für Shard-Verteilung.

ObjectGridQuery
ObjectGrid-Abfrage.

ObjectGridReplication
Replikationsservice.

ObjectGridRouting
Details zum Client/Server-Routing.

ObjectGridSecurity
Sicherheits-Trace.

ObjectGridStats
ObjectGrid-Statistiken.

ObjectGridStreamQuery
Die API ″Stream Query″.

ObjectGridWriteBehind
ObjectGrid-Write-Behind.

Projector
Die Steuerkomponente in der API ″EntityManager″.

QueryEngine
Die Abfragesteuerkomponente für die API ″Object Query″ und die API ″EntityManager Query″.

QueryEnginePlan
Diagnose des Abfrageplans.

Kapitel 9. Implementierungsumgebung sichern

Zum Schutz Ihrer Daten in WebSphere eXtreme Scale kann eXtreme Scale mit externen Sicherheitsprovidern integriert werden.

WebSphere eXtreme Scale kann mit einer externen Sicherheitsimplementierung integriert werden. Diese externe Implementierung muss Authentifizierungs- und Berechtigungsservices für eXtreme Scale bereitstellen. eXtreme Scale hat Plug-in-Punkte für die Integration einer Sicherheitsimplementierung. eXtreme Scale wurde erfolgreich in die folgenden Komponenten integriert:

- Lightweight Directory Access Protocol (LDAP)
- Kerberos
- ObjectGrid-Sicherheit
- Tivoli Access Manager
- Java Authentication and Authorization Service (JAAS)

eXtreme Scale verwendet den Sicherheitsprovider für die folgenden Tasks:

- Authentifizierung von Clients bei Servern,
- Berechtigung von Clients für den Zugriff auf bestimmte Artefakte von eXtreme Scale oder Festlegung der Verwendung von eXtreme-Scale-Artefakten.

eXtreme Scale hat die folgenden Typen von Berechtigungen:

Map-Berechtigung
: Clients oder Gruppen können für die Durchführung von Einfüge-, Lese-, Aktualisierungs- oder Löschoperationen in Maps berechtigt werden.

ObjectGrid-Berechtigung
: Clients oder Gruppen können für die Ausführung von Objekt- oder Entitätsabfragen in ObjectGrids berechtigt werden.

DataGrid-Agentenberechtigung
: Clients oder Gruppen können für die Implementierung von DataGrid-Agenten in ein ObjectGrid berechtigt werden.

Serverseitige Map-Berechtigung
: Clients oder Gruppen können für die Replikation einer Server-Map auf Clientseite oder die Erstellung eins dynamischen Index für die Server-Map berechtigt werden.

Verwaltungsberechtigung
: Clients oder Gruppen können für die Ausführung von Verwaltungs-Tasks berechtigt werden.

Anmerkung: Wenn Sie die Sicherheit für Ihr Back-End bereits aktiviert haben, müssen Sie beachten, dass diese Sicherheitseinstellungen für den Schutz Ihrer Daten nicht mehr ausreichen. Die Sicherheitseinstellungen Ihrer Datenbank oder eines anderen Datenspeichers werden in keiner Weise auf Ihren Cache übertragen. Sie müssen die Daten, die jetzt zwischengespeichert werden, mit dem Sicherheitsmechanismus von eXtreme Scale schützen, der Authentifizierung, Berechtigung und Sicherheit auf Transportebene umfasst.

Grid-Sicherheit

Die Grid-Sicherheit von WebSphere eXtreme Scale gewährleistet, dass ein dem Grid beitretender Server die richtigen Berechtigungsnachweise besitzt, so dass kein zerstörerischer Server in das Grid aufgenommen wird. Zu diesem Zweck wird ein Mechanismus mit Shared-Secret-Zeichenfolgen verwendet.

Alle Server von WebSphere eXtreme Scale, einschließlich der Katalogserver, einigen sich auf eine Shared-Secret-Zeichenfolge. Wenn ein Server dem Grid beitritt, wird er aufgefordert, diese Shared-Secret-Zeichenfolge vorzulegen. Wenn die Shared-Secret-Zeichenfolge des beitretenden Servers mit der Zeichenfolge im führenden Server oder Katalogserver übereinstimmt, wird der beitretende Server akzeptiert, wenn nicht, wird die Join-Anforderung zurückgewiesen.

Das Senden einer Shared-Secret-Zeichenfolge als Klartext ist nicht sicher. Die Sicherheitsinfrastruktur von WebSphere eXtreme Scale stellt ein Manager-Plug-in für sichere Token bereit, damit der Server dieses Shared Secret vor dem Senden sichern kann. Sie müssen festlegen, wie die Sicherungsoperation implementiert wird. WebSphere eXtreme Scale stellt eine vordefinierte Implementierung bereit, in der die Sicherungsoperation so implementiert ist, dass das Shared Secret verschlüsselt und signiert wird.

Die Shared-Secret-Zeichenfolge wird in der Datei `server.properties` festgelegt. Weitere Informationen zur Eigenschaft ″authenticationSecret″ finden Sie im Abschnitt „Servereigenschaftendatei” auf Seite 197.

SecureTokenManager-Plug-in

Ein Manager-Plug-in für sichere Token wird über die Schnittstelle ″com.ibm.websphere.objectgrid.security.plugins.SecureTokenManager″ bereitgestellt.

Weitere Informationen zum SecureTokenManager-Plug-in finden Sie in der Dokumentation zur API ’SecureTokenManager’.

Die Methode ″generateToken(Object)″ verwendet ein Objekt und generiert anschließend ein Token, das von anderen nicht interpretiert werden kann. Die Methode ″verifyTokens(byte[])″ führt den umgekehrten Prozess aus. Sie konvertiert das Token zurück in das ursprüngliche Objekt.

Eine einfache SecureTokenManager-Implementierung verwendet einen einfachen Verschlüsselungsalgorithmus, wie z. B. einen XOR-Algorithmus (exklusives Oder), um das Objekt in serialisierter Form zu verschlüsseln, und anschließend den entsprechenden Entschlüsselungsalgorithmus, um das Token zu entschlüsseln. Diese Implementierung ist nicht sicher.

WebSphere eXtreme Scale stellt eine sofort verfügbare Implementierung für diese Schnittstelle bereit.

Die Standardimplementierung verwendet ein Schlüsselpaar, um die Signatur zu signieren und zu prüfen, und einen geheimen Schlüssel, um den Inhalt zu verschlüsseln. Jeder Server hat einen JCKES-Keystore, in dem das Schlüsselpaar (privater und öffentlicher Schlüssel) und der geheime Schlüssel gespeichert werden. Der Keystore muss ein JCKES-Keystore sein, damit geheime Schlüssel gespeichert werden können.

Diese Schlüssel werden verwendet, um die Shared-Secret-Zeichenfolge auf Senderseite zu verschlüsseln und zu signieren bzw. zu prüfen. Außerdem wird dem Token eine Verfallszeit zugeordnet. Auf Empfängerseite werden die Daten geprüft, entschlüsselt und mit der Shared-Secret-Zeichenfolge des Empfängers verglichen. Es sind keine SSL-Kommunikationsprotokolle (Secure Sockets Layer) zwischen einem Serverpaar für die Authentifizierung erforderlich, weil die privaten und öffentlichen Schlüssel demselben Zweck dienen. Wenn die Serverkommunikation jedoch nicht verschlüsselt ist, können die Daten einfach durch Ansicht der Kommunikation gestohlen werden. Da das Token relativ bald verfällt, ist das Sicherheitsrisiko durch Attacken durch Nachrichtenaufzeichnung und -wiederholung minimal. Das Risiko ist erheblich geringer, wenn alle Server hinter einer Firewall implementiert werden.

Dieser Ansatz hat den Nachteil, dass die eXtreme-Scale-Administratoren Schlüssel generieren und an alle Server übermitteln müssen, was während des Transports der Schlüssel zu Sicherheitsverletzungen führen kann.

Konfiguration

Weitere Informationen zu den Eigenschaften, mit denen Sie den Manager für sichere Token konfigurieren, finden Sie im Abschnitt Servereigenschaften.

Lokale Sicherheit aktivieren

WebSphere eXtreme Scale stellt mehrere Sicherheitsendpunkte für die Integration angepasster Mechanismen bereit. Im lokalen Programmiermodell ist die Hauptsicherheitsfunktion Berechtigung. Authentifizierung wird nicht unterstützt. Sie müssen die Authentifizierung unabhängig von der bereits vorhandenen Authentifizierung in WebSphere Application Server durchführen. Es werden jedoch Plug-ins für das Anfordern und Validieren von Subject-Objekten bereitgestellt.

In den folgenden Abschnitten werden zwei Methoden beschrieben, mit denen Sie die lokale Sicherheit aktivieren können:

Mit einer ObjectGrid-XML-Datei können Sie eine ObjectGrid-Instanz definieren und die Sicherheit für diese Instanz aktivieren. Die Datei `secure-objectgrid-definition.xml`, die in der Musterunternehmensanwendung ″ObjectGridSample″ verwendet wird, wird im folgenden Beispiel gezeigt. Setzen Sie das Attribut ″securityEnabled″ auf `true`, um die Sicherheit zu aktivieren.

```
<objectGrids>
    <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
        authorizationMechanism="AUTHORIZATION_MECHANISM_JAAS">
    ...
</objectGrids>
```

Sie können die Sicherheit auch über das Programm aktivieren. Zum Erstellen eines ObjectGrids mit der Methode ″ObjectGrid.setSecurityEnabled″ rufen Sie die folgende Methode in der Schnittstelle ″ObjectGrid″ auf:

```
/**
 * ObjectGrid-Sicherheit aktivieren
 */
void setSecurityEnabled();
```

Weitere Informationen finden Sie in der API-Dokumentation.

Anwendungsclientauthentifizierung

Die Anwendungsclientauthentifizierung setzt sich aus der Aktivierung der Client/Server-Sicherheit, der Authentifizierung des Berechtigungsnachweises und der Konfiguration eines Authentifikators und eines Generators für Systemberechtigungsnachweise zusammen.

Client/Server-Sicherheit aktivieren

Für eine erfolgreiche Authentifizierung bei ObjectGrid müssen Sie die Sicherheit im Client und im Server aktivieren.

Clientsicherheit aktivieren

WebSphere eXtreme Scale stellt eine Musterclienteigenschaftendatei mit dem Namen `sampleClient.properties` im Verzeichnis *WAS_HOME*`/optionalLibraries/ObjectGrid/properties` für eine Installation von WebSphere Extended Deployment bzw. im Verzeichnis `/ObjectGrid/properties` für eine heterogene Serverinstallation bereit. Sie können diese Schablonendatei mit entsprechenden Werten anpassen. Setzen Sie die Eigenschaft "securityEnabled" in der Datei `objectgridClient.properties` auf `true`. Die Eigenschaft "securityEnabled" gibt an, ob die Sicherheit aktiviert ist. Wenn ein Client eine Verbindung zu einem Server herstellt, muss die Eigenschaft "securityEnabled" auf der Client- und auf der Serverseite denselben Wert haben: `true` oder `false`. Ist die Sicherheit beispielsweise im verbindungsherstellenden Server aktiviert, muss die Eigenschaft im Client auf `true` gesetzt werden, damit die Verbindung zum Server hergestellt werden kann.

Die Schnittstelle "com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration" stellt die Datei `security.ogclient.props` dar. Sie können die allgemein zugängliche Anwendungsprogrammierschnittstelle "com.ibm.websphere.objectgrid.security.config.ClientSecurityConfigurationFactory" verwenden, um eine Instanz dieser Schnittstelle mit Standardwerten zu erstellen. Sie können aber auch eine Instanz erstellen, indem Sie die Datei mit den Sicherheitseigenschaften des ObjectGrid-Clients übergeben. Die Datei `security.ogclient.props` enthält weitere Eigenschaften. Weitere Einzelheiten finden Sie in der Dokumentation zur API "ClientSecurityConfiguration" und in der Dokumentation zur API "ClientSecurityConfigurationFactory".

Serversicherheit aktivieren

Zum Aktivieren der Sicherheit auf der Serverseite können Sie die Eigenschaft "securityEnabled" in der Datei `security.xml` auf `true` setzen. Verwenden Sie eine XML-Sicherheitsdeskriptordatei, um die Grid-Sicherheitskonfiguration so zu definieren, das die Grid-weite Sicherheitskonfiguration von der Konfiguration, die sich nicht auf die Sicherheit bezieht, isoliert wird.

Authentifizierung des Berechtigungsnachweises

Nachdem der eXtreme-Scale-Client das Credential-Objekt mit dem CredentialGenerator-Objekt abgerufen hat, wird das Credential-Objekt zusammen mit der Clientanforderung an den eXtreme-Scale-Server gesendet. Der Server authentifiziert das Credential-Objekt, bevor er die Anforderung verarbeitet. Bei erfolgreicher Authentifizierung des Credential-Objekts wird ein Subject-Objekt zurückgegeben, dass dieses Credential-Objekt repräsentiert. Dieses Subject-Objekt wird anschließend für die Berechtigung der Anforderung verwendet.

Setzen Sie ″credentialAuthentication″ in den Client- und Servereigenschaftendateien so, dass die Authentifizierung des Berechtigungsnachweises aktiviert wird. Weitere Informationen finden Sie in den Abschnitten „Clienteigenschaftendatei” auf Seite 203 und „Servereigenschaftendatei” auf Seite 197.

Die folgende Tabelle enthält eine Übersicht über die für verschiedene Einstellungen zu verwendenden Authentifizierungsverfahren.

Tabelle 9. Authentifizierung des Berechtigungsnachweises bei Client- und Servereinstellungen

Authentifizierung des Clientberechtigungsnachweises	Authentifizierung des Serverberechtigungsnachweises	Ergebnis
Nein	Nie	Inaktiviert
Nein	Unterstützt	Inaktiviert
Nein	Erforderlich	Fehlersituation
Unterstützt	Nie	Inaktiviert
Unterstützt	Unterstützt	Aktiviert
Unterstützt	Erforderlich	Aktiviert
Erforderlich	Nie	Fehlersituation
Erforderlich	Unterstützt	Aktiviert
Erforderlich	Erforderlich	Aktiviert

Authentifikator konfigurieren

Der eXtreme-Scale-Server verwendet das Authenticator-Plug-in, um das Credential-Objekt zu authentifizieren. Eine Implementierung der Schnittstelle ″Authenticator″ ruft das Credential-Objekt ab und authentifiziert es dann bei einer Benutzer-Registry, z. B. einem LDAP-Server (Lightweight Directory Access Protocol) usw. eXtreme Scale stellt keine Registry-Konfiguration bereit. Die Herstellung einer Verbindung zu und die Authentifizierung bei einer Benutzer-Registry muss in diesem Plug-in implementiert werden.

Eine Authenticator-Implementierung extrahiert beispielsweise die Benutzer-ID und das Kennwort aus dem Berechtigungsnachweis, verwendet diese Informationen für die Herstellung einer Verbindung zu einem und Validierung bei einem LDAP-Server und erstellt als Ergebnis der Authentifizierung ein Subject-Objekt. Die Implementierung kann JAAS-Anmeldemodule (Java Authentication and Authorization Service) verwenden. Als Ergebnis der Authentifizierung wird ein Subject-Objekt zurückgegeben.

Sie können den Authentifikator, wie im folgenden Beispiel gezeigt, in der XML-Sicherheitsdeskriptordatei konfigurieren:

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

 <security securityEnabled="true" loginSessionExpirationTime="300" >

<authenticator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.KeyStoreLoginAuthenticator">
        </authenticator>

 </security>

</securityConfig>
```

Verwenden Sie die Option ″–clusterSecurityFile″ beim Starten eines sicheren Servers, um die XML-Sicherheitsdatei festzulegen. Weitere Informationen finden Sie im Lernprogramm zur Java-SE-sicherheit im *Produktübersicht*.

Generator für Systemberechtigungsnachweise konfigurieren

Der Generator für Systemberechtigungsnachweise wird für die Darstellung einer Factory für die Systemberechtigungsnachweise verwendet. Ein Systemberechtigungsnachweis gleicht einem Administratorberechtigungsnachweis. Sie können das Element ″SystemCredentialGenerator″ in der XML für Katalogsicherheit konfigurieren, wie es im folgenden Beispiel gezeigt wird:

```
<systemCredentialGenerator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator">
          <property name="properties" type="java.lang.String" value="manager manager1" description="username password" />
    </systemCredentialGenerator>
```

Zu Demonstrationszwecken werden Benutzername und Kennwort in Klartext gespeichert. Speichern Sie den Benutzernamen und das Kennwort in einer Produktionsumgebung nicht in Klartext.

WebSphere eXtreme Scale stellt einen Standardgenerator für Systemberechtigungsnachweise bereit, der die Serverberechtigungsnachweise verwendet. Wenn Sie den Generator für Systemberechtigungsnachweise nicht explizit angeben, wird dieser Standardgenerator für Systemberechtigungsnachweise verwendet.

Anwendungsclientberechtigung

Die Anwendungsclientberechtigung setzt sich aus ObjectGrid-Berechtigungsklassen, Berechtigungsmechanismen, einem Berechtigungsprüfintervall und dem Feature ″Zugriff nur durch Ersteller″ zusammen.

Bei eXtreme Scale basiert die Berechtigung auf dem Subject-Objekt und auf Berechtigungen. Das Produkt unterstützt zwei Arten von Berechtigungsmechanismen: Java Authentication and Authorization Service (JAAS) und angepasste Berechtigungen.

ObjectGrid-Berechtigungsklassen

Die Berechtigung basiert auf Berechtigungen. Es gibt die folgenden vier verschiedenen Typen von Berechtigungsklassen:

- Die Klasse ″MapPermission″ stellt Berechtigungen für den Zugriff auf die Daten in ObjectGrid-Maps dar.
- Die Klasse ″ObjectGridPermission″ stellt Berechtigungen für den Zugriff auf ObjectGrid dar.
- Die Klasse ″ServerMapPermission″ stellt Berechtigungen für den Zugriff auf ObjectGrid-Maps auf der Serverseite über einen Client dar.
- Die Klasse ″AgentPermission″ stellt Berechtigungen zum Starten eines Agenten auf der Serverseite dar.

Weitere Informationen zu APIs und zugehörigen Berechtigungen finden Sie im Abschnitt zur Programmierung der Clientberechtigung im *Programmierhandbuch*.

Berechtigungsprüfintervall

eXtreme Scale unterstützt das Caching der Berechtigungsprüfergebnisse für Leistungszwecke. Ist dieser Mechanismus nicht vorhanden und wird eine in der Liste der Methoden und der erforderlichen Berechtigungen aufgeführte Methode

aufgerufen, ruft die Laufzeitumgebung den konfigurierten Berechtigungsmechanismus zur Berechtigung des Zugriffs auf. Wenn das Berechtigungsprüfintervall definiert ist, wird der Berechtigungsmechanismus auf der Basis des festgelegten Berechtigungsprüfintervalls in regelmäßigen Abständen aufgerufen.

Die Berechtigungsinformationen für die Berechtigungen basieren auf dem Subject-Objekt. Wenn ein Client versucht, auf die Methoden zuzugreifen, durchsucht die Laufzeitumgebung von eXtreme Scale den Cache nach dem Subject-Objekt. Wird das Objekt nicht im Cache gefunden, prüft die Laufzeitumgebung die für dieses Subject-Objekt erteilten Berechtigungen und speichert dann die Berechtigungen in einem Cache.

Das Berechtigungsprüfintervall muss vor der Initialisierung von ObjectGrid definiert werden. Sie können das Berechtigungsprüfintervall auf zwei Arten konfigurieren:

Sie können die ObjectGrid-XML-Datei verwenden, um ein ObjectGrid zu definieren und das Berechtigungsprüfintervall festzulegen. Im folgenden Beispiel wird das Berechtigungsprüfintervall auf 45 Sekunden gesetzt:

```
<objectGrids>
 <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
 authorizationMechanism="AUTHORIZATION_MECHANISM_JAAS"
 permissionCheckPeriod="45">
  <bean id="bean id="TransactionCallback"
className="com.ibm.websphere.samples.objectgrid.HeapTransactionCallback" />
...
</objectGrids>
```

Wenn Sie ein ObjectGrid über APIs erstellen möchten, rufen Sie die folgende Methode auf, um das Berechtigungsprüfintervall festzulegen. Diese Methode kann nur vor der Initialisierung der ObjectGrid-Instanz aufgerufen werden. Sie gilt nur dann für das lokale eXtreme-Scale-Programmiermodell, wenn Sie die ObjectGrid-Instanz direkt instanzieren.

```
/**
 * Diese Methode akzeptiert einen einzigen Parameter, der angibt, wie oft
 * die Berechtigung geprüft werden soll, die verwendet wird, um einem
 * Clientzugriff zuzulassen. Wenn der Parameter den Wert 0 hat, wird
 * der Berechtigungsmechanismus (JAAS-Berechtigung oder angepasste
 * Berechtigung) bei jedem Aufruf von get/put/update/remove/evict
 * aufgefordert zu prüfen, ob das aktuelle Subject-Objekt die erforderlichen
 * Berechtigungen besitzt. Dies kann je nach Berechtigungsimplementierung
 * vom Leistungsstandpunkt aus gesehen untragbar kostenintensiv sein, aber
 * sollten Sie den Berechtigungsmechanismus jemals aufrufen müssen, dann
 * setzen Sie den Parameter auf 0.
* Wenn Sie den Parameter auf einen Wert > 0 setzen, gibt er an, wie lange
 * (in Sekunden) ein Berechtigungssatz zwischengespeichert werden soll, bevor
 * sie zur Aktualisierung an den Berechtigungsmechanismus zurückgegeben werden.
 * Dieser Wert liefert eine sehr viel bessere Leistung, aber wenn die
 * Back-End-Berechtigungen in dieser Zeit geändert werden, kann das
 * ObjectGrid den Zugriff zulassen oder verweigern, obwohl der
 * Back-End-Sicherheitsprovider geändert wurde.
*
 * @param Zeitraum - Das Berechtigungsprüfintervall in Sekunden
 */
void setPermissionCheckPeriod(int period);
```

Berechtigung ″Zugriff nur durch Ersteller″

Die Berechtigung ″Zugriff nur durch Ersteller″ gewährleistet, dass ausschließlich der Benutzer (dargestellt durch die zugeordneten Principal-Objekte), der den Eintrag in die ObjectGrid-Map einfügt, auf diesen Eintrag zugreifen (lesen, aktualisieren, ungültig machen und entfernen) kann.

Das vorhandene Berechtigungsmodell für ObjectGrid-Maps basiert auf dem Zugriffstyp, aber nicht auf Dateneinträgen. Anders ausgedrückt, ein Benutzer kann mit einem bestimmten Zugriffstyp (z. B. lesen, schreiben, einfügen, löschen oder ungültig machen) entweder auf alle Daten in der Map oder auf keine Daten in der

Map zugreifen. eXtreme Scale berechtigt Benutzer jedoch nicht für den Zugriff auf einzelne Dateneinträge. Dieses Feature bietet eine neue Methode für die Berechtigung von Benutzern für den Zugriff auf Dateneinträge.

In einem Szenario, in dem verschiedene Benutzer auf verschiedene Datengruppen zugreifen, kann dieses Modell hilfreich sein. Wenn der Benutzer Daten aus dem persistenten Speicher in die ObjectGrid-Maps lädt, kann der Zugriff vom persistenten Speicher berechtigt werden. In diesem Fall muss keine weitere Berechtigung auf der Ebene der ObjectGrid-Maps erfolgen. Sie müssen lediglich sicherstellen, dass die Person, die die Daten in die Map lädt, auf die Map zugreifen kann, indem Sie das Feature ″Zugriff nur durch Ersteller″ aktivieren.

Es gibt drei verschiedene Modi für das Feature ″Zugriff nur durch Ersteller″:

disabled
Das Feature ″Zugriff nur durch Ersteller″ ist inaktiviert.

complement
Das Feature ″Zugriff nur durch Ersteller″ ist aktiviert, um die Map-Berechtigung zu ergänzen. In anderen Worten, die Map-Berechtigung und das Feature ″Zugriff nur durch Ersteller″ sind wirksam. Deshalb können Sie die Datenoperationen weiter einschränken. Der Ersteller kann die Daten beispielsweise nicht ungültig machen.

supersede
Das Feature ″Zugriff nur durch Ersteller″ ist aktiviert, um die Map-Berechtigung außer Kraft zu setzen. In anderen Worten, das Feature ″Zugriff nur durch Ersteller″ setzt die Map-Berechtigung außer Kraft, d. h., es wird keine Map-Berechtigung durchgeführt.

Sie können den Modus für das Feature ″Zugriff nur durch Ersteller″ auf zwei Arten konfigurieren:

Sie können die ObjectGrid-XML-Datei verwenden, um ein ObjectGrid zu definieren und den Modus für das Feature ″Zugriff nur durch Ersteller″ auf `disabled` (Inaktiviert), `complement` (Ergänzung) oder `supersede` (Überlagern) zu setzen,

```
<objectGrids>
    <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
         accessByCreatorOnlyMode="supersede"
        <bean id="TransactionCallback"
              classname="com.ibm.websphere.samples.objectgrid.HeapTransactionCallback" />
    ...
</objectGrids>
```

Wenn Sie ein ObjectGrid über das Programm erstellen möchten, können Sie die folgende Methode aufrufen, um den Modus für das Feature ″Zugriff nur durch Ersteller″ festzulegen. Der Aufruf dieser Methode gilt nur dann für das lokale eXtreme-Scale-Programmiermodell, wenn Sie die ObjectGrid-Instanz direkt instanzieren:

```
/**
 * Legen Sie den Modus für das Feature "Zugriff nur durch Ersteller"
 * (ACCESS_BY_CREATOR_ONLY) fest.
 * Wenn Sie das Feature "Zugriff nur durch Ersteller" aktivieren, kann nur
 * der Benutzer (dargestellt durch die zugeordneten Principals), der den
 * Datensatz in die Map einfügt, auf den Datensatz zugreifen (lesen,
 * aktualisieren, ungültig machen und entfernen).
* Das Feature "Zugriff nur durch Ersteller"  kann inaktiviert werden
 * oder das ObjectGrid-Berechtigungsmodell ergänzen oder sogar außer
 * Kraft setzen. Standardmäßig ist das Feature inaktiviert:
 * {@link SecurityConstants#ACCESS_BY_CREATOR_ONLY_DISABLED}.
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_DISABLED
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_COMPLEMENT
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_SUPERSEDE
 *
 * @param accessByCreatorOnlyMode the access by creator mode.
```

```
 *
 * @since WAS XD 6.1 FIX3
*/
void setAccessByCreatorOnlyMode(int accessByCreatorOnlyMode);
```

Zur weiteren Veranschaulichung stellen Sie sich ein Szenario vor, in dem eine ObjectGrid-Map ″account″ ein Banken-Grid ist und Manager1 und Employee1 zwei Benutzer sind. Die eXtreme-Scale-Berechtigungsrichtlinie erteilt ″Manager1″ alle Zugriffsberechtigungen, ″Employee1″ aber nur Lesezugriff. Die JAAS-Richtlinie für die ObjectGrid-Map-Berechtigung ist im folgenden Beispiel gezeigt:

```
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
    Principal com.acme.PrincipalImpl "Manager1" {
    permission com.ibm.websphere.objectgrid.security.MapPermission
        "banking.account", "all"
};
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
    Principal com.acme.PrincipalImpl "Employee1" {
    permission com.ibm.websphere.objectgrid.security.MapPermission
        "banking.account", "read"
};
```

Machen Sie sich Gedanken darüber, wie sich das Feature ″Zugriff nur durch Ersteller″ auf die Berechtigung auswirkt:

- **Inaktiviert:** Wenn das Feature ″Zugriff nur durch Ersteller″ inaktiviert ist, sind keine Auswirkungen auf die Map-Berechtigung zu verzeichnen. Der Benutzer ″Manager1″ kann auf alle Daten in der Map ″account″ zugreifen. Der Benutzer ″Employee1″ kann alle Daten in der Map lesen, aber keine Daten in der Map aktualisieren, ungültig machen oder entfernen.
- **Ergänzung:** Wenn Sie das Feature ″Zugriff nur durch Ersteller″ mit der Option ″complement″ (Ergänzung) aktivieren, sind die Map-Berechtigung und die Berechtigung über das Feature ″Zugriff nur durch Ersteller″ wirksam. Der Benutzer ″Manager1″ kann auf die Daten in der Map ″account″ zugreifen, aber nur dann, wenn der Benutzer allein sie in die Map geladen hat. Der Benutzer ″Employee1″ kann die Daten in der Map ″account″ lesen, aber nur dann, wenn dieser Benutzer allein sie in die Map geladen hat. (Dieser Benutzer kann jedoch keine Daten in der Map aktualisieren, ungültig machen oder entfernen.)
- **Überlagern:** Wenn Sie das Feature ″Zugriff nur durch Ersteller″ mit der Option ″supersede″ (Überlagern) aktivieren, wird die Map-Berechtigung nicht umgesetzt. Die Berechtigung über das Feature ″Zugriff nur durch Ersteller″ ist die einzige Berechtigungsrichtlinie. Der Benutzer ″Manager1″ hat dieselben Privilegien wie im Modus ″Ergänzung″. Er kann nur auf die Daten in der Map ″account″ zugreifen, wenn er selbst die Daten in die Map geladen hat. Der Benutzer ″Employee1″ hat jedoch vollständigen Zugriff auf die Daten in der Map ″account″, wenn er selbst die Daten in die Map geladen hat. Anders ausgedrückt, die in der JAAS-Richtlinie definierte Berechtigungsrichtlinie wird nicht umgesetzt.

Transport Layer Security und Secure Sockets Layer

WebSphere eXtreme Scale unterstützt TCP/IP und Transport Layer Security/Secure Sockets Layer (TLS/SSL) für die sichere Kommunikation zwischen Client und Server in dynamischen Implementierungsmodellen.

TLS/SSL unterstützt die sichere Kommunikation zwischen Client und Server. Der verwendete Kommunikationsmechanismus richtet sich nach dem Wert des Parameters ″transportType″, der in den Konfigurationsdateien von Client und Server angegeben ist.

Sie können die Eigenschaft ″transportType″ in den folgenden Client- und Serverkonfigurationsdateien definieren.

- Informationen zum Definieren der Eigenschaft in der Clientsicherheitskonfiguration finden Sie im Abschnitt „Clienteigenschaftendatei" auf Seite 203.
- Informationen zum Definieren der Eigenschaft in der Sicherheitskonfiguration des Containerservers finden Sie im Abschnitt „Servereigenschaftendatei" auf Seite 197.
- Informationen zum Definieren der Eigenschaft in der Sicherheitskonfiguration des Katalogservers finden Sie im Abschnitt „Servereigenschaftendatei" auf Seite 197.

Tabelle 10. Für bestimmte Clienttransport- und Servertransporteinstellungen zu verwendendes Protokoll

Eigenschaft "transportType" des Clients	Eigenschaft "transportType" des Servers	Zu verwendendes Protokoll
TCP/IP	TCP/IP	TCP/IP
TCP/IP	SSL-supported	TCP/IP
TCP/IP	SSL-required	Fehler
SSL-supported	TCP/IP	TCP/IP
SSL-supported	SSL-supported	SSL (wenn SSL scheitert, dann TCP/IP)
SSL-supported	SSL-required	SSL
SSL-required	TCP/IP	Fehler
SSL-required	SSL-supported	SSL
SSL-required	SSL-required	SSL

Wenn SSL verwendet wird, müssen die SSL-Konfigurationsparameter auf Client- und auf Serverseite angegeben werden. In einer Java-SE-Umgebung wird SSL in den Client- und Servereigenschaftendateien konfiguriert. Wenn der Client oder Server in WebSphere Application Server ausgeführt wird, können Sie die Unterstützung für Transportsicherheit von WebSphere Application Server für die Konfiguration der SSL-Parameter verwenden.

Datei `orb.properties` für die Unterstützung der Transportsicherheit konfigurieren

Sie können TLS/SSL verwenden, wenn die Eigenschaft "transportType" den Wert "SSL-Supported" hat.

Zur Unterstützung sicherer Transporte in einer Java-SE-Umgebung müssen Sie die Datei „ORB-Eigenschaftendatei" auf Seite 207 ändern und die folgenden Eigenschaften einfügen:

```
# Eigenschaften des IBM JDK
org.omg.CORBA.ORBClass=com.ibm.CORBA.iiop.ORB
org.omg.CORBA.ORBSingletonClass=com.ibm.rmi.corba.ORBSingleton
javax.rmi.CORBA.StubClass=com.ibm.rmi.javax.rmi.CORBA.StubDelegateImpl
javax.rmi.CORBA.PortableRemoteObjectClass=com.ibm.rmi.javax.rmi.PortableRemoteObject
javax.rmi.CORBA.UtilClass=com.ibm.ws.orb.WSUtilDelegateImpl

# WS-Plug-ins
com.ibm.CORBA.ORBPluginClass.com.ibm.ws.orbimpl.transport.WSTransport
com.ibm.CORBA.ORBPluginClass.com.ibm.ws.orbimpl.WSORBPropertyManager
com.ibm.CORBA.ORBPluginClass.com.ibm.ISecurityUtilityImpl.SecurityPropertyManager

# WS-Interceptor
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.objectgrid.corba.ObjectGridInitializer
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ISecurityLocalObjectBaseL13Impl.SecurityComponentFactory

# Eigenschaften des WS-ORB und der Plug-ins
com.ibm.ws.orb.transport.ConnectionInterceptorName=com.ibm.ISecurityLocalObjectBaseL13Impl.SecurityConnectionInterceptor
com.ibm.ws.orb.transport.WSSSLClientSocketFactoryName=com.ibm.ws.security.orbssl.WSSSLClientSocketFactoryImpl

com.ibm.CORBA.TransportMode=Pluggable
com.ibm.CORBA.ServerName=ogserver
```

SSL-Parameter für eXtreme-Scale-Clients konfigurieren

Sie können SSL-Parameter für Clients wie folgt konfigurieren:

1. Erstellen Sie mit der Factory-Klasse ″com.ibm.websphere.objectgrid.security.config. ClientSecurityConfigurationFactory″ ein com.ibm.websphere.objectgrid.security.config.SSLConfiguration-Objekt. Weitere Einzelheiten finden Sie in der API-Dokumentation zu ClientSecurityConfigurationFactory.
2. Konfigurieren Sie die Parameter in der Datei `client.propeties`, und verwenden Sie anschließend die Methode ″ClientSecurityConfigurationFactory.getClientSecurityConfiguration(String)″, um die Objektinstanz mit Daten zu füllen.

Beispiele für Eigenschaften, die Sie in einem Client definieren können, finden Sie in der Beschreibung der Sicherheitseigenschaften des Clients im Abschintt „Clienteigenschaftendatei″ auf Seite 203.

SSL-Parameter für eXtreme-Scale-Server konfigurieren

Die SSL-Parameter für Server werden in einer Servereigenschaftendatei wie der Beispieldatei `server.properties` konfiguriert. Diese Eigenschaftendatei kann beim Starten eines eXtreme-Scale-Servers als Parameter übergeben werden. Weitere Informationen zu den SSL-Parametern, die Sie für eXtreme-Scale-Server definieren können finden Sie im Abschnitt „Servereigenschaftendatei″ auf Seite 197.

Unterstützung der Transportsicherheit in WebSphere Application Server

Wenn ein Client, Containerserver oder Katalogserver von eXtreme Scale in einem Prozess von WebSphere Application Server ausgeführt wird, wird die Transportsicherheit von eXtreme Scale über die CSIV2-Transporteinstellungen des Anwendungsservers verwaltet. In diesem Fall dürfen Sie nicht die Eigenschaften des eXtreme-Scale-Clients oder -Servers verwenden, um die SSL-Einstellungen zu konfigurieren. Alle SSL-Einstellungen müssen in der Konfiguration von WebSphere Application Server definiert werden.

Anmerkung:

Wenn Sie Ihre SSL-Einstellungen mit einer Eigenschaftendatei konfigurieren, überschreiben Sie die vorhandenen Einstellungen.

Zugehörige Verweise

„Referenz der Eigenschaftendatei" auf Seite 196
Servereigenschaftendateien enthalten Einstellungen für die Ausführung der Katalogserver und Containerserver. Sie können eine Servereigenschaftendatei für eine eigenständige Konfiguration oder eine Konfiguration von WebSphere Application Server angeben. Clienteigenschaftendateien enthalten Einstellungen für Ihren Client.

„Servereigenschaftendatei" auf Seite 197
Die Servereigenschaftendatei enthält verschiedene Eigenschaften, mit denen die verschiedenen Einstellungen für Ihren Server definiert werden, z. B. Trace-Einstellungen, Protokollierung und Sicherheitskonfiguration. Die Servereigenschaftendatei wird vom Katalogservice und von den Containerservern verwendet.

„Clienteigenschaftendatei" auf Seite 203
Sie können eine Eigenschaftendatei erstellen, die auf Ihren Anforderungen für eXtreme-Scale-Clientprozesse basiert.

„ORB-Eigenschaftendatei" auf Seite 207
Die Datei `orb.properties` wird für die Übergabe der vom Object Request Broker (ORB) verwendeten Eigenschaften verwendet, um das Transportverhalten des Grids zu ändern.

Grid-Authentifizierung

Das sichere Tokenmanager-Plug-in ist ein weiteres Plug-in, das für die Authentifizierung zwischen Servern verwendet wird. Das sichere Tokenmanager-Plug-in wird durch die Schnittstelle "com.ibm.websphere.objectgrid.security.plugins.SecureTokenManager" dargestellt.

Die Methode "generateToken(Object)" verwendet ein Objekt und generiert anschließend ein Token, das von anderen nicht interpretiert werden kann. Die Methode "verifyTokens(byte[])" führt den umgekehrten Prozess aus. Sie konvertiert das Token zurück in das ursprüngliche Objekt.

Eine einfache SecureTokenManager-Implementierung verwendet einen einfachen Verschlüsselungsalgorithmus, wie z. B. einen XOR-Algorithmus (exklusives Oder), um das Objekt in serialisierter Form zu verschlüsseln, und anschließend den entsprechenden Entschlüsselungsalgorithmus, um das Token zu entschlüsseln. Diese Implementierung ist nicht sicher und anfällig für Attacken.

Standardimplementierung von WebSphere eXtreme Scale

WebSphere eXtreme Scale stellt eine sofort verfügbare Implementierung für diese Schnittstelle bereit. Diese Standardimplementierung verwendet ein Schlüsselpaar, um die Signatur zu signieren und zu prüfen, und einen geheimen Schlüssel, um den Inhalt zu verschlüsseln. Jeder Server hat einen JCKES-Keystore, in dem das Schlüsselpaar (privater und öffentlicher Schlüssel) und der geheime Schlüssel gespeichert werden. Der Keystore muss ein JCKES-Keystore sein, damit geheime Schlüssel gespeichert werden können. Diese Schlüssel werden verwendet, um die Shared-Secret-Zeichenfolge auf Senderseite zu verschlüsseln und zu signieren bzw. zu prüfen. Außerdem wird dem Token eine Verfallszeit zugeordnet. Auf Empfängerseite werden die Daten geprüft, entschlüsselt und mit der Shared-Secret-Zeichenfolge des Empfängers verglichen. Es sind keine SSL-Kommunikationsprotokolle (Secure Sockets Layer) zwischen einem Serverpaar für die Authentifizierung erforderlich, weil die privaten und öffentlichen Schlüssel demselben Zweck dienen. Wenn die Serverkommunikation jedoch nicht verschlüsselt ist, können die Daten einfach durch Ansicht der Kommunikation gestohlen werden. Da das Token

relativ bald verfällt, ist das Sicherheitsrisiko durch Attacken durch Nachrichtenaufzeichnung und -wiederholung minimal. Das Risiko ist erheblich geringer, wenn alle Server hinter einer Firewall implementiert werden.

Dieser Ansatz hat den Nachteil, dass die Administratoren von WebSphere eXtreme Scale Schlüssel generieren und an alle Server übermitteln müssen, was während des Transports der Schlüssel zu Sicherheitsverletzungen führen kann.

JMX-Sicherheit (Java Management Extensions)

Sie können MBean-Aufrufe (You Beans) in einer dynamischen Implementierungsumgebung sichern.

Weitere Informationen zu den verfügbaren MBeans finden Sie im Abschnitt „Managed Beans (MBeans) für die Verwaltung der Umgebung verwenden" auf Seite 285.

In der dynamischen Implementierungstopologie befinden sich MBeans direkt in den Katalogservern und Containerservern. Im Allgemeinen folgt die JMX-Sicherheit in einer dynamischen Implementierungstopologie der JMX-Sicherheitsspezifikation, die in der Spezifikation "Java Management Extensions" festgelegt ist. Sie setzt sich aus den folgenden drei Komponenten zusammen:

1. Authentifizierung: Der ferne Client muss im Connector-Server authentifiziert werden.
2. Zugriffssteuerung: Die MBean-Zugriffssteuerung legt fest, wer auf die MBean-Informationen zugreifen und wer die MBean-Operationen durchführen kann.
3. Sicherer Transport: Der Transport zwischen dem JMX-Client und dem Server kann mit TLS/SSL gesichert werden.

Authentifizierung

JMX stellt den Connector-Servern Methoden für die Authentifizierung der fernen Clients zur Verfügung. Für den RMI-Connector wird die Authentifizierung durchgeführt, indem beim Erstellen des Connector-Servers ein Objekt übergeben wird, das die Schnittstelle "JMXAuthenticator" implementiert. Deshalb implementiert eXtreme Scale die Schnittstelle "JMXAuthenticator", um das ObjectGrid-Authenticator-Plug-in zu nutzen, um die fernen Clients zu authentifizieren. In dem Lernprogramm zur Sicherheit im Handbuch *Produktübersicht* wird detailliert beschrieben, wie eXtreme Scale einen Client authentifiziert.

Der JMX-Client folgt den JMX-APIs, um die Berechtigungsnachweise für die Verbindungsherstellung zum Connector-Server bereitzustellen. Das JMX-Framework übergibt die Berechtigungsnachweise an den Connector-Server und ruft dann die JMXAuthenticator-Implementierung für die Authentifizierung auf. Wie zuvor beschrieben, delegiert die JMXAuthenticator-Implementierung die Authentifizierung an die ObjectGrid-Authenticator-Implementierung.

Sehen Sie sich das folgende Beispiel an, das veranschaulicht, wie mit einem Berechtigungsnachweis eine Verbindung zu einem Connector-Server hergestellt wird:

```
javax.management.remote.JMXServiceURL jmxUrl = new JMXServiceURL(
        "service:jmx:rmi:///jndi/rmi://localhost:1099/objectgrid/MBeanServer");

    environment.put(JMXConnector.CREDENTIALS, new UserPasswordCredential("admin", "xxxxxx"));

    // JMXConnectorServer erstellen
```

```
JMXConnector cntor = JMXConnectorFactory.newJMXConnector(jmxUrl, null);

// Verbindung herstellen und eine Operation im fernen MBeanServer aufrufen
cntor.connect(environment);
```

Im vorherigen Beispiel wird ein UserPasswordCredential-Objekt mit der Benutzer-ID ″admin″ und dem Kennwort ″xxxxx″ bereitgestellt. Dieses UserPasswordCredential-Objekt wird in der Umgebungs-Map gesetzt, die von der Methode ″JMXConnector.connect(Map)″ verwendet wird. Anschließend wird dieses UserPasswordCredential-Objekt über das JMX-Framework zunächst an den Server und schließlich zur Authentifizierung an das ObjectGrid-Authentifizierungs-Framework übergeben.

Das Clientprogrammiermodell folgt strikt der JMX-Spezifikation.

Zugriffssteuerung

Ein JMX-MBean-Server kann Zugriff auf sensible Informationen haben und deshalb in der Lage sein, sensible Operationen durchzuführen. JMX stellt die erforderliche Zugriffssteuerung bereit, die feststellt, welche Clients auf diese Informationen zugreifen und welche Clients diese Operationen durchführen dürfen. Die Zugriffssteuerung wird in das Java-Standardsicherheitsmodell integriert, indem Berechtigungen definiert werden, die den Zugriff auf den MBean-Server und seine Operationen steuern.

Bei der Zugriffssteuerung und -berechtigung für JMX-Operationen stützt sich eXtreme Scale auf die JAAS-Unterstützung, die die JMX-Implementierung bereitstellt. Zu jedem beliebigen Zeitpunkt während der Ausführung eines Programms besitzt ein Ausführungs-Thread einen aktuellen Satz an Berechtigungen. Wenn ein solcher Thread eine Operation der JMX-Spezifikation aufruft, handelt es sich um die so genannten ″gehaltenen Berechtigungen″. Bei der Durchführung einer JMX-Operation wird eine Sicherheitsprüfung durchgeführt, um festzustellen, ob die ″gehaltenen Berechtigungen″ die erforderliche Berechtigung abdecken.

Die MBean-Richtliniendefinition folgt dem Java-Richtlinienformat. Die folgende Richtlinie erteilt beispielsweise allen Unterzeichnern und allen Codebasen das Recht, die JMX-Adresse des Servers für die MBean ″PlacementServiceMBean″ abzurufen, allerdings mit Einschränkung auf die Domäne ″com.ibm.websphere.objectgrid″:

```
grant {
    permission javax.management.MBeanPermission
        "com.ibm.websphere.objectgrid.management.PlacementServiceMBean#retrieveServerJMXAddress
     [com.ibm.websphere.objectgrid:*,type=PlacementService]",
        "invoke";
}
```

Sie können das folgende Richtlinienbeispiel verwenden, um die Berechtigung auf der Basis der Identität des fernen Clients durchzuführen. Die Richtlinie erteilt dieselben MBean-Berechtigungen wie im vorherigen Beispiel, aber nur den Benutzern mit dem X500Principal-Namen ″CN=Administrator,OU=software,O=IBM,L=Rochester,ST=MN,C=US″.

```
grant principal javax.security.auth.x500.X500Principal "CN=Administrator,OU=software,O=IBM,L=Rochester,ST=MN,C=US" {
    permission javax.management.MBeanPermission
        "com.ibm.websphere.objectgrid.management.PlacementServiceMBean#retrieveServerJMXAddress
    [com.ibm.websphere.objectgrid:*,type=PlacementService]",
        "invoke";
}
```

Java-Richtlinien werden nur geprüft, wenn der Sicherheitsmanager aktiviert ist. Starten Sie Katalogserver und Containerserver mit dem JVM-Argument ″-Djava.security.manager″, um die Zugriffssteuerung für MBean-Operationen umzusetzen.

Sicherer Transport

Der Transport zwischen dem JMX-Client und dem Server kann mit TLS/SSL gesichert werden. Wenn das Attribut ″transportType″ des Katalogserver oder Containerservers auf ″SSL_Required″ oder ″SSL_Supported″ gesetzt ist, müssen Sie SSL verwenden, um die Verbindung zum JMX-Server herzustellen.

Zur Verwendung von SSL müssen Sie den Truststore, den Truststore-Typ und das Truststore-Kennwort im MBean-Client mit Systemeigenschaften konfigurieren, die mit ″-D″ beginnen:

1. -Djavax.net.ssl.trustStore=TRUST_STORE_LOCATION
2. -Djavax.net.ssl.trustStorePassword=TRUST_STORE_PASSWORD
3. -Djavax.net.ssl.trustStoreType=TRUST_STORE_TYPE

Wenn Sie ″com.ibm.websphere.ssl.protocol.SSLSocketFactory″ als SSL-Socket-Factory in der Datei ″JAVA_HOME/jre/lib/security/java.security″ verwenden, definieren Sie die folgenden Eigenschaften:

1. -Dcom.ibm.ssl.trustStore=TRUST_STORE_LOCATION
2. -Dcom.ibm.ssl.trustStorePassword=TRUST_STORE_PASSWORD
3. -Dcom.ibm.ssl.trustStoreType=TRUST_STORE_TYPE

XML-Sicherheitsdeskriptordatei

Verwenden Sie eine ObjectGrid-XML-Sicherheitsdeskriptordatei, um eine Implementierungstopologie von eXtreme Scale mit aktivierter Sicherheit zu konfigurieren. In diesem Abschnitt werden verschiedene Konfigurationen anhand von XML-Musterdateien veranschaulicht.

Jedes Element und Attribut der XML-Datei wird in der folgenden Liste beschrieben. Verwenden Sie die Beispiele, um sich mit der Verwendung dieser Elemente und Attribute für die Konfiguration der Umgebung vertraut zu machen.

Element ″securityConfig″

Das Element ″securityConfig″ ist das Ausgangselement der ObjectGrid-XML-Sicherheitsdatei. Dieses Element konfiguriert den Namespace der Datei und die Schemaposition. Das Schema wird in der Datei `objectGridSecurity.xsd` definiert.

- Anzahl der Vorkommen: 1
- Untergeordnete Elemente: security

Element ″security″

Verwenden Sie das Element ″security″, um die Sicherheit eines ObjectGrids zu definieren.

- Anzahl der Vorkommen: 1
- Untergeordnete Elemente: authenticator, adminAuthorization und systemCredentialGenerator

Attribute

securityEnabled
Aktiviert die Sicherheit für das Grid, wenn es den Wert ″true″ hat. Der Standardwert ist ″false″. Wenn das Attribut den Wert ″false″ hat, ist die Grid-weite

Sicherheit inaktiviert. Weitere Informationen finden Sie im Abschnitt „Grid-Sicherheit" auf Seite 310. (Optional)

singleSignOnEnabled
Wenn Sie das Attribut auf "true" setzen, kann ein Client eine Verbindung zu jedem Server herstellen, nachdem er bei einem der Server authentifiziert wurde. Hat das Attribut den Wert "false", muss ein Client sich bei jedem Server authentifizieren, bevor er eine Verbindung herstellen kann. Der Standardwert ist "false". (Optional)

loginSessionExpirationTime
Gibt die Verfallszeit der Anmeldesitzung in Sekunden an. Wenn die Anmeldesitzung abläuft, muss sich der Client erneut authentifizieren. (Optional)

adminAuthorizationEnabled
Aktiviert die Verwaltungsberechtigung. Wenn dieses Attribut den Wert "true" hat, müssen alle Verwaltungs-Tasks berechtigt werden. Der verwendete Berechtigungsmechanismus wird mit dem Wert des Attributs "adminAuthorizationMechanism" angegeben. Der Standardwert ist "false". (Optional)

adminAuthorizationMechanism
Gibt an, welcher Berechtigungsmechanismus zu verwenden ist. WebSphere eXtreme Scale unterstützt zwei Berechtigungsmechanismen: Java Authentication and Authorization Service (JAAS) und angepasste Berechtigung. Der Berechtigungsmechanismus JAAS verwendet den richtlinienbasierten JAAS-Standardansatz. Wenn Sie JAAS als Berechtigungsmechanismus festlegen möchten, setzen Sie das Attribut auf AUTHORIZATION_MECHANISM_JAAS. Der angepasste Berechtigungsmechanismus verwendet eine vom Benutzer integrierte AdminAuthorization-Implementierung. Wenn Sie einen angepassten Berechtigungsmechanismus angeben möchten, setzen Sie das Attribut auf AUTHORIZATION_MECHANISM_CUSTOM. Weitere Informationen zur Verwendung dieser beiden Mechanismen finden Sie im Abschnitt „Anwendungsclientberechtigung" auf Seite 314. (Optional)

Die folgende Datei `security.xml` ist eine Musterkonfiguration zum Aktivieren der Sicherheit eines eXtreme-Scale-Grids:

security.xml

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

    <security securityEnabled="true" singleSignOnEnabled="true"
        loginSessionExpirationTime="20"
        adminAuthorizationEnabled="true"
        adminAuthorizationMechanism="AUTHORIZATION_MECHANISM_JAAS" >

        <authenticator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator">
        </authenticator>

        <systemCredentialGenerator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator">
            <property name="properties" type="java.lang.String" value="runAs" description="Using runAs subject" />
        </systemCredentialGenerator>

     </security></securityConfig>
```

Element "authenticator"

Authentifiziert Clients bei eXtreme-Scale-Servern im Grid. Die Klasse, die mit dem Attribut "className" angegeben wird, muss die Schnittstelle "com.ibm.websphere.objectgrid.security.plugins.Authenticator" implementieren. Der Authentifikator kann Eigenschaften verwenden, um Methoden in der Klasse aufzurufen, die mit dem Attribut "className" angegeben wird. Weitere Informationen zur Verwendung von Eigenschaften finden Sie in der Beschreibung des Elements "property".

In der vorherigen Beispieldatei `security.xml` wurde die Klasse ″com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator″ als Authentifikator angegeben. Diese Klasse implementiert die Schnittstelle ″com.ibm.websphere.objectgrid.security.plugins.Authenticator″.

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: property

Attribute

className
Gibt eine Klasse an, die die Schnittstelle ″com.ibm.websphere.objectgrid.security.plugins.Authenticator″ implementiert. Verwenden Sie diese Klasse, um Clients bei Servern im eXtreme-Scale_Grid zu authentifizieren. (Erforderlich)

Element ″adminAuthorization″

Verwenden Sie das Element ″adminAuthorization″, um den Verwaltungszugriff auf das Grid zu konfigurieren.

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: property

Attribute

className
Gibt eine Klasse an, die die Schnittstelle ″com.ibm.websphere.objectgrid.security.plugins.AdminAuthorization″ implementiert. (Erforderlich)

Element ″systemCredentialGenerator″

Verwenden Sie ein Element ″systemCredentialGenerator″, um einen Generator für Systemberechtigungsnachweise zu konfigurieren. Dieses Element gilt nur für eine dynamische Umgebung. Im dynamischen Konfigurationsmodell stellt der dynamische Containerserver als eXtreme-Scale-Client eine Verbindung zum Katalogserver her, und der Katalogserver kann ebenfalls als Client eine Verbindung zum eXtreme-Scale-Containerserver herstellen. Dieser Generator für Systemberechtigungsnachweise wird für die Darstellung einer Factory für die Systemberechtigungsnachweise verwendet.

- Anzahl der Vorkommen: 0 oder 1
- Untergeordnetes Element: property

Attribute

className
Gibt eine Klasse an, die die Schnittstelle ″com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator″ implementiert. (Erforderlich)

Ein Beispiel zur Verwendung des Elements ″systemCredentialGenerator″ entnehmen Sie der vorherigen Datei `security.xml`. In diesem Beispiel ist com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenCredentialGenerator der Generator für Systemberechtigungsnachweise, der das RunAs-Subject-Objekt aus dem Thread abruft.

Element ″property″

Ruft set-Methoden in den Klassen ″authenticator″ und ″adminAuthorization″ auf. Der Name der Eigenschaft entspricht einer set-Methode im Attribut ″className″ des Elements ″authenticator″ bzw. ″adminAuthorization″.

- Anzahl der Vorkommen: 0 oder mehr
- Untergeordnetes Element: property

Attribute

name
Der Name der Eigenschaft. Der Wert, der diesem Attribut zugeordnet wird, muss einer set-Methode in der Klasse entsprechen, die mit dem Attribut ″class-Name″ der übergeordneten Bean angegeben wird. Wenn das Attribut ″class-Name″ der Bean beispielsweise auf ″com.ibm.MyPlugin″ gesetzt ist und der Name der angegebenen Eigenschaft ″size″ lautet, muss die Klasse ″com.ibm-.MyPlugin″ eine Methode ″setSize″ haben. (Erforderlich)

type
Gibt den Typ der Eigenschaft an. Der Typ des Parameters, der an die mit dem Attribut ″name″ angegebene set-Methode übergeben wird. Die gültigen Werte sind primitive Java-Typen, ihre java.lang-Gegenstücke und java.lang.String. Die Attribute ″name″ und ″type″ müssen einer Methodensignatur im Attribut ″className″ der Bean entsprechen. Wenn der Name beispielsweise ″size″ und der Typ ″int″ ist, muss eine Methode ″setSize(int)″ in der Klasse vorhanden sein, die mit dem Attribut ″className″ für die Bean angegeben wurde. (Erforderlich)

value
Gibt den Wert der Eigenschaft an. Dieser Wert wird in den mit dem Attribut ″type″ angegebenen Typ konvertiert und anschließend als Parameter in dem Aufruf der set-Methode verwendet, die mit den Attributen ″name″ und ″type″ angegeben wurde. Der Wert dieses Attributs wird nicht validiert. Der Plug-in-Implementierer muss sicherstellen, dass der übergebene Wert gültig ist. (Erforderlich)

description
Gibt eine Beschreibung der Eigenschaft an. (Optional)

Weitere Informationen finden Sie im Abschnitt „Datei `objectGridSecurity.xsd`″ auf Seite 195.

Integration der Sicherheit mit WebSphere Application Server

WebSphere eXtreme Scale stellt mehrere Sicherheitsfeatures für die Integration in die Sicherheitsstruktur von WebSphere Application Server bereit.

Integration der Authentifizierung

Wenn eXtreme-Scale-Clients und -Server in WebSphere Application Server und in derselben Sicherheitsdomäne ausgeführt werden, können Sie die Sicherheits-infrastruktur von WebSphere Application Server verwenden, um die Berechtigungsnachweise für die Clientauthentifizierung an den eXtreme-Scale-Server weiterzugeben. Versucht ein Servlet beispielsweise als eXtreme-Scale-Client eine Verbindung zu einem eXtreme-Scale-Server in derselben Sicherheitsdomäne herzustellen und ist das Servlet bereits authentifiziert, kann das Authentifizierungs-token von Client (Servlet) an den Server weitergegeben und anschließend die

Sicherheitsinfrastruktur von WebSphere Application Server verwendet werden, um das Authentifizierungstoken an die Clientberechtigungsnachweise zurückzugeben.

Integration der Sicherheit für verteilte Umgebungen mit WebSphere Application Server

Für das verteilte ObjectGrid-Modell kann die Integration der Sicherheit über die folgenden Klassen vorgenommen werden:

```
com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator
com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator
com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredential
```

Weitere Informationen finden Sie im Abschnitt „Anwendungsclientauthentifizierung" auf Seite 312. Das folgende Beispiel veranschaulicht, wie die Klasse "WSTokenCredentialGenerator" verwendet wird:

```
/**
 * Verbindung zum ObjectGrid-Server herstellen
 */
protected ClientClusterContext connect() throws ConnectException {
  ClientSecurityConfiguration csConfig = ClientSecurityConfigurationFactory
  .getClientSecurityConfiguration(proFile);

  CredentialGenerator gen = getWSCredGen();

  csConfig.setCredentialGenerator(gen);

  return objectGridManager.connect(csConfig, null);
}

/**
 * WSTokenCredentialGenerator abrufen
 *
private CredentialGenerator getWSCredGen() {
    WSTokenCredentialGenerator gen = new WSTokenCredentialGenerator(
        WSTokenCredentialGenerator.RUN_AS_SUBJECT);
    return gen;
}
```

Verwenden Sie serverseitig den WSTokenAuthentication-Authentifikator, um das WSTokenCredential-Objekt zu authentifizieren.

Integration der Sicherheit für lokale Umgebungen mit WebSphere Application Server

Für das lokale ObjectGrid-Modell kann die Integration der Sicherheit über die folgenden beiden Klassen vorgenommen werden:

- com.ibm.websphere.objectgrid.security.plugins.builtins.WSSubjectSourceImpl
- com.ibm.websphere.objectgrid.security.plugins.builtins.WSSubjectValidationImpl

Weitere Informationen zu diesen Klassen finden Sie in der Beschreibung der lokalen Sicherheit im *Programmierhandbuch*. Sie können die Klasse "WSSubjectSourceImpl" als SubjectSource-Plug-in und die Klasse "WSSubjectValidationImpl" als SubjectValidation-Plug-in konfigurieren.

Sichere eXtreme-Scale-Server starten und stoppen

Servers müssen für die Implementierungsumgebung häufig sicher sein, und dies erfordert eine spezielle Konfiguration.

Sicheren Server in einer Java-SE-Umgebung starten

Sie können einen Katalogservice oder Containerserver wie folgt starten.

Sicheren eXtreme-Scale-Katalogservice starten

Für das Starten eines sicheren eXtreme-Scale-Katalogserviceprozesses sind zwei weitere Sicherheitskonfigurationsdateien erforderlich:

XML-Sicherheitsdeskriptordatei: Die XML-Sicherheitsdeskriptordatei beschreibt die Sicherheitseigenschaften, die für alle Server (einschließlich Katalogservern und Containerservern) gelten. Ein Beispiel für eine solche Eigenschaft ist die Authentifikatorkonfiguration, die die Benutzer-Registry und das Authentifizierungsverfahren darstellt.

Servereigenschaftendatei. Die Servereigenschaftendatei konfiguriert die für den Server spezifischen Sicherheitseigenschaften.

Wenn Sie den Befehl ″startOgServer.sh″ bzw. ″startOgServer.cat″ zum Starten eines sicheren eXtreme-Scale-Katalogserviceprozesses verwenden, können Sie ″-clusterSecurityFile″ oder ″-clusterSecurityUrl″ verwenden, um die XML-Sicherheitsdeskriptordatei als Dateityp oder URL-Typ festzulegen, und Sie können ″–serverProps″ verwenden, um die Servereigenschaftendatei zu setzen.

Sicheren eXtreme-Scale-Containerserver starten

Für das Starten eines sicheren eXtreme-Scale-Containerservers ist eine einzige Sicherheitskonfigurationsdatei erforderlich:

- **Servereigenschaftendatei:** Die Servereigenschaftendatei konfiguriert die für den Server spezifischen Sicherheitseigenschaften. Weitere Einzelheiten finden Sie im Abschnitt „Servereigenschaftendatei” auf Seite 197.

Wenn Sie den Befehl ″startOgServer.sh″ bzw. ″startOgServer.cat″ zum Starten eines sicheren eXtreme-Scale-Containerservers verwenden, können Sie mit ″–serverProps″ die Servereigenschaftendatei festlegen. Es gibt weitere Methoden zum Festlegen der Servereigenschaftendatei. Weitere Informationen hierzu finden Sie in der Beschreibung der Servereigenschaftendatei.

Einzelheiten zur Verwendung des Befehls startOgServer.sh bzw. startOgServer.bat und der zugehörigen Optionen finden Sie im Abschnitt „Script ″startOgServer″” auf Seite 235.

Sicheren eXtreme-Scale-Server stoppen

Für das Stoppen eines sicheren eXtreme-Scale-Katalogserviceprozesses bzw. Containerservers ist eine einzige Sicherheitskonfigurationsdatei erforderlich:

- **Clienteigenschaftendatei:** Die Clienteigenschaftendatei kann zum Konfigurieren der Clientsicherheitseigenschaften verwendet werden. Die Clientsicherheitseigenschaften sind erforderlich, damit ein Client eine Verbindung zu einem sicheren Server herstellen kann. Weitere Einzelheiten finden Sie im Abschnitt „Clienteigenschaftendatei” auf Seite 203.

Wenn Sie den Befehl ″stopOgServer.sh″ bzw. ″stopOgServer.cat″ zum Stoppen eines sicheren eXtreme-Scale-Katalogserviceprozesses oder -Containerservers verwenden, können Sie mit ″-clientSecurityFile″ die Sicherheitseigenschaften des Clients festlegen.

Einzelheiten zur Verwendung des Befehls ″stopOgServer.sh″ bzw. ″stopOgServer-.cat″ und der zugehörigen Optionen finden Sie im Abschnitt „Script ″stopOgServer″″ auf Seite 241.

Sicheren Server in WebSphere Application Server starten

Das Starten eines sicheren ObjectGrid-Servers in WebSphere Application Server verläuft ähnlich wie das Starten eines nicht sicheren ObjectGrid-Servers, abgesehen davon, dass Sie die Sicherheitskonfigurationsdateien übergeben müssen. Anstatt wie in der J2SE-Umgebung das Argument ″-[EIGENSCHAFTENDATEI]″ (z. B. -serverProps) im Befehl zu übergeben, verwenden Sie das Argument ″-D[EIGENSCHAFTENDATEI] in den generischen JVM-Argumenten.

Sicheren Katalogservice in WebSphere Application Server starten

Ein Katalogserver enthält zwei verschiedene Stufen von Sicherheitsinformationen:

- `-Dobjectgrid.cluster.security.xml.url`: Gibt die Datei ″objectGridSecurity.xml″ an, die die Sicherheitseigenschaften beschreibt, die für alle Server (einschließlich Katalogservern und Containerservern) gelten. Ein Beispiel ist die Authentifikatorkonfiguration, die die Benutzer-Registry und das Authentifizierungsverfahren darstellt. Der Dateiname für diese Eigenschaft muss im URL-Format angegeben werden, z. B. ″file:///tmp/og/objectGridSecurity.xml″.
- `-Dobjectgrid.server.props`: Gibt die Servereigenschaftendatei an, die die serverspezifischen Sicherheitseigenschaften enthält. Der für diese Eigenschaft angegebene Dateiname kann im herkömmlichen Dateipfadformat angegeben werden, z. B. ″c:/tmp/og/catalogserver.props″. Die Verwendung von ″-Dobjectgrid.security-.server.props″ ist veraltet, aber Sie können diese Option für die Abwärtskompatibilität weiterhin verwenden.

Zum Starten eines sicheren Katalogservice in WebSphere Application Server folgen Sie den Anweisungen im Absatz ″Integriert in WebSphere Application Server″ des Abschnitts „Grid-Sicherheit” auf Seite 310.

Als nächstes definieren Sie die Sicherheitseigenschaft in den generischen JVM-Argumenten des Prozesses:

```
-Dobjectgrid.cluster.security.xml.url=file:///tmp/og/
objectGridSecurity.xml-Dobjectgrid.server.props=/tmp/og/
catalog.server.props
```

Im Folgenden wird beschrieben, wie Sie die Eigenschaft den generischen JVM-Argumenten hinzufügen:

- Erweitern Sie den Eintrag ″Systemverwaltung″ in der Task-Ansicht auf der linken Seite.
- Klicken Sie auf den Prozess von WebSphere Application Server, in dem der Katalogservice implementiert ist, z. B. ″Deployment Manager″.
- Erweitern Sie auf der rechten Seite den Eintrag ″Java- und Prozessverwaltung″ unter ″″Serverinfrastruktur″.
- Klicken Sie auf ″Prozessdefinition″.

- Klicken Sie auf ″Java Virtual Machine″ unter ″Weitere Eigenschaften″.
- Geben Sie die Eigenschaften im Textfeld ″Generische JVM-Argumente″ ein.

Sicheren Containerserver in WebSphere Application Server starten

Ein Containerserver erhält bei der Verbindungsherstellung zum Katalogserver alle Sicherheitskonfigurationen, die in der Datei ″objectGridSecurity.xml″ konfiguriert sind, wie z. B. die Authentifikatorkonfiguration oder das Zeitlimit für Anmeldesitzungen. Außerdem muss ein Containerserver seine eigenen serverspezifischen Sicherheitseigenschaften in der Eigenschaft ″-Dobjectgrid.server.props″ konfigurieren.

Die Eigenschaft ″-Dobjectgrid.server.props″ wird an Stelle der Eigenschaft ″-Dobjectgrid.security.server.props″ verwendet, weil in diese Eigenschaftendatei auch andere nicht sicherheitsbezogene Eigenschaften aufgenommen werden. Der Dateiname für diese Eigenschaft kann im herkömmlichen Dateipfadformat angegeben werden, z. B. `c:/tmp/og/server.props`.

Führen Sie dieselben Schritte wie zuvor aus, um die Sicherheitseigenschaft den generischen JVM-Argumenten hinzuzufügen.

Zugehörige Tasks

„Eigenständige Server von WebSphere eXtreme Scale starten” auf Seite 228
Wenn Sie eine eigenständige Konfiguration von WebSphere eXtreme Scale verwenden, setzt sich die Umgebung aus Katalogservern, Containerservern und eXtreme-Scale-Clientprozessen zusammen. Sie müssen diese Prozesse manuell konfigurieren und starten.

„Katalogservice in einer eigenständigen Umgebung starten” auf Seite 229
Sie müssen den Katalogservice manuell starten, wenn Sie eine verteilte Umgebung von WebSphere eXtreme Scale ohne WebSphere Application Server verwenden.

„Containerprozesse starten” auf Seite 232
Sie können eXtreme Scale über die Befehlszeile, über eine Implementierungstopologie oder über eine Datei `server.properties` starten.

Zugehörige Verweise

„Script ″startOgServer″” auf Seite 235
Das Script ″startOgServer″ startet Server. Sie können beim Starten Ihrer Server eine Vielzahl von Parametern verwenden, um die Trace-Erstellung zu aktivieren, Portnummern anzugeben usw.

„Protokolle und Trace” auf Seite 303
Sie können Protokolle und Trace verwenden, um Ihre Umgebung zu überwachen und Fehler zu beheben. Protokolle werden je nach Konfiguration an unterschiedlichen Positionen gespeichert. Möglicherweise müssen Sie einen Trace für einen Server bereitstellen, wenn Sie mit der IBM Unterstützungsfunktion zusammenarbeiten.

Kapitel 10. Fehlerbehebung

Zusätzlich zu den Protokollen und Traces, Nachrichten und Releaseinformationen können Sie Überwachungstools verwenden, um Fehler in Ihrer Konfiguration zu beheben.

Überwachungstools für die Fehlerbehebung verwenden

Zusätzlich zu Protokollen, Trace, Nachrichten und Releaseinformationen können Sie Überwachungstools verwenden, um Gegebenheiten zu verstehen, wie z. B. die Position der Daten in der Umgebung, die Verfügbarkeit der Server im Grid usw. Wenn Sie in einer Umgebung mit WebSphere Application Server arbeiten, können Sie Performance Monitoring Infrastructure (PMI) verwenden. Wenn Sie ein einer eigenständigen Umgebung arbeiten, können Sie Überwachungstools anderer Anbieter verwenden, wie z. B. CA Wily Introscope oder Hyperic HQ. Außerdem können Sie das Musterdienstprogramm ″xsAdmin″ verwenden und anpassen, um Textinformationen zu Ihrer Umgebung anzuzeigen.

Weitere Informationen zu Überwachungstools finden Sie im Abschnitt Kapitel 8, „Implementierungsumgebung überwachen”, auf Seite 263.

Protokolle und Trace

Sie können Protokolle und Trace verwenden, um Ihre Umgebung zu überwachen und Fehler zu beheben. Protokolle werden je nach Konfiguration an unterschiedlichen Positionen gespeichert. Möglicherweise müssen Sie einen Trace für einen Server bereitstellen, wenn Sie mit der IBM Unterstützungsfunktion zusammenarbeiten.

Protokolle mit WebSphere Application Server

Weitere Informationen finden Sie im Information Center von WebSphere Application Server.

Protokolle mit WebSphere eXtreme Scale in einer eigenständigen Umgebung

Bei eigenständigen Katalog- und Containerservern legen Sie die Position der Protokolle und alle Trace-Spezifikationen fest. Die Katalogserverprotokolle werden dort gespeichert, wo Sie den Befehl zum Starten des Servers ausführen.

Protokollposition für Containerserver festlegen

Standardmäßig werden die Protokolle für einen Server in dem Verzeichnis gespeichert, in dem der Befehl zum Starten des Servers ausgeführt wird. Wenn Sie die Server im Verzeichnis *<Ausgangsverzeichnis_von_eXtreme_Scale>*/bin starten, werden die Protokoll- und Trace-Dateien in den Verzeichnissen logs/*<Servername>* des Verzeichnisses bin gespeichert. Wenn Sie eine andere Position für die Protokolle eines Containerservers festlegen möchten, erstellen Sie eine Eigenschaftendatei, z. B. server.properties, mit dem folgenden Inhalt:

```
workingDirectory=<Verzeichnis>
traceSpec=
systemStreamToFileEnabled=true
```

Die Eigenschaft ″workingDirectory″ ist das Stammverzeichnis für die Protokolle und die optionale Trace-Datei. WebSphere eXtreme Scale erstellt ein Verzeichnis mit dem Namen des Containerservers mit einer Datei `SystemOut.log`, einer Datei `SystemErr.log` und einer Trace-Datei, falls die Trace-Erstellung mit der Option ″traceSpec″ aktiviert wurde. Wenn eine Eigenschaftendatei während des Containerstarts verwendet werden soll, verwenden Sie die Option **-serverProps**, und geben Sie die Position der Servereigenschaftendatei an.

Weitere Informationen finden Sie in den Abschnitten „Eigenständige Server von WebSphere eXtreme Scale starten″ auf Seite 228 und „Script ″startOgServer″″ auf Seite 235.

Häufige Informationsnachrichten, die in der Datei `SystemOut.log` aufgezeichnet werden, sind Bestätigungsnachrichten für den Start. Weitere Informationen zu bestimmten Nachrichten finden Sie im Abschnitt „Nachrichten″ auf Seite 335.

Trace-Erstellung mit WebSphere Application Server

Weitere Informationen finden Sie im Information Center von WebSphere Application Server.

Trace-Erstellung für den Katalogservice

Sie können die Trace-Erstellung für einen Katalogservice festlegen, indem Sie während des Katalogservicestarts die Parameter **-traceSpec** und **-traceFile** verwenden. Beispiel:

```
startOgServer.sh catalogServer -traceSpec
ObjectGridPlacement=all=enabled -traceFile
/home/user1/logs/trace.log
```

Wenn Sie den Katalogservice im Verzeichnis `<Ausgangsverzeichnis_von_eXtreme_Scale>/bin` starten, werden die Protokolle und Trace-Dateien in einem Verzeichnis `logs/<Name_des_Katalogservice>` des Verzeichnisses `bin` gespeichert. Weitere Einzelheiten zum Starten eines Katalogservice finden Sie im Abschnitt „Katalogservice in einer eigenständigen Umgebung starten″ auf Seite 229.

Trace für einen eigenständigen Containerserver erstellen

Sie können die Trace-Erstellung für eine Containerserver auf zwei Arten aktivieren. Sie können, wie im Abschnitt zu den Protokollen erläutert, eine Servereigenschaftendatei erstellen, oder Sie können die Trace-Erstellung beim Start über die Befehlszeile aktivieren. Zum Aktivieren der Trace-Erstellung für den Container über eine Servereigenschaftendatei aktualisieren Sie die Eigenschaft **traceSpec** mit der erforderlichen Trace-Spezifikation. Zum Aktivieren der Trace-Erstellung für den Container über Startparameter verwenden Sie die Parameter **-traceSpec** und **-traceFile**. Beispiel:

```
startOgServer.sh c0 -objectGridFile ../xml/myObjectGrid.xml
-deploymentPolicyFile ../xml/myDepPolicy.xml -catalogServiceEndpoints
server1.rchland.ibm.com:2809 -traceSpec
ObjectGridPlacement=all=enabled -traceFile /home/user1/logs/trace.log
```

Wenn Sie den Server im Verzeichnis `<Ausgangsverzeichnis_von_eXtreme_Scale>/bin` starten, werden die Protokoll- und Trace-Dateien in den Verzeichnissen `logs/<Servername>` des Verzeichnisses `bin` gespeichert. Weitere Einzelheiten zum Starten eines Containerprozesses finden Sie im Abschnitt „Containerprozesse starten″ auf Seite 232.

Trace-Erstellung mit der Schnittstelle ″ObjectGridManager″

Eine weitere Option ist die Definition der Trace-Erstellung zur Laufzeit über eine ObjectGridManager-Schnittstelle. Die Definition der Trace-Erstellung in einer ObjectGridManager-Schnittstelle kann verwendet werden, um einen Trace für einen eXtreme-Scale-Client zu erstellen, wenn dieser eine Verbindung zu einer eXtreme-Scale-Instanz herstellt und Transaktionen festschreibt. Wenn Sie die Trace-Erstellung in einer ObjectGridManager-Schnittstelle festlegen möchten, geben Sie eine Trace-Spezifikation und ein Trace-Protokoll an.

```
ObjectGridManager manager = ObjectGridManagerFactory.getObjectGridManager();
...
manager.setTraceEnabled(true);
manager.setTraceFileName("logs/myClient.log");
manager.setTraceSpecification("ObjectGridReplication=all=enabled");
```

Trace-Erstellung mit dem Dienstprogramm ″xsadmin″ aktivieren

Zum Aktivieren der Trace-Erstellung mit dem Dienstprogramm ″xsadmin″ verwenden Sie die Option **setTraceSpec**. Verwenden Sie das Dienstprogramm ″xsadmin″, um die Trace-Erstellung für eine eigenständige Umgebung zur Laufzeit und nicht zur Startzeit zu aktivieren. Sie können die Trace-Erstellung für alle Server und Katalogservices aktivieren, oder Sie können die Server nach dem ObjectGrid-Namen usw. filtern. Wenn Sie beispielsweise den ObjectGridReplication-Trace mit Zugriff auf den Katalogserviceserver aktivieren möchten, führen Sie den folgenden Befehl aus:

```
<Ausgangsverzeichnis_von_eXtreme_Scale>/bin>xsadmin.bat -setTraceSpec "ObjectGridReplication=all=enabled"
```

Sie können die Trace-Erstellung auch inaktivieren, indem Sie die Trace-Spezifikation auf `*=all=disabled.` setzen.

Weitere Informationen finden Sie im Abschnitt „Musterdienstprogramm ″xsAdmin″ verwenden” auf Seite 287.

FFDC-Verzeichnis und -Dateien

FFDC-Dateien sind als Debug-Hilfe für die IBM Unterstützungsfunktion bestimmt. Diese Dateien werden möglicherweise von der IBM Unterstützungsfunktion angefordert, wenn ein Problem auftritt.

Diese Dateien befinden sich in einem Verzeichnis mit dem Namen `ffdc`. Das Verzeichnis enthält Dateien wie die folgenden:

```
server2_exception.log
server2_20802080_07.03.05_10.52.18_0.txt
```

Zugehörige Konzepte

„Sichere eXtreme-Scale-Server starten und stoppen" auf Seite 327
Servers müssen für die Implementierungsumgebung häufig sicher sein, und dies erfordert eine spezielle Konfiguration.

Zugehörige Tasks

„Eigenständige Server von WebSphere eXtreme Scale starten" auf Seite 228
Wenn Sie eine eigenständige Konfiguration von WebSphere eXtreme Scale verwenden, setzt sich die Umgebung aus Katalogservern, Containerservern und eXtreme-Scale-Clientprozessen zusammen. Sie müssen diese Prozesse manuell konfigurieren und starten.

„Katalogservice in einer eigenständigen Umgebung starten" auf Seite 229
Sie müssen den Katalogservice manuell starten, wenn Sie eine verteilte Umgebung von WebSphere eXtreme Scale ohne WebSphere Application Server verwenden.

„Containerprozesse starten" auf Seite 232
Sie können eXtreme Scale über die Befehlszeile, über eine Implementierungstopologie oder über eine Datei `server.properties` starten.

Trace-Optionen

Sie können die Trace-Erstellung aktivieren, um der IBM Unterstützungsfunktion Informationen über Ihre Umgebung bereitzustellen.

Informationen zur Trace-Erstellung

Der WebSphere eXtreme Scale-Trace ist in mehrere Komponenten unterteilt. Ähnlich wie bei der Trace-Erstellung in WebSphere Application Server können Sie die zu verwendende Trace-Stufe angeben. Zu den gebräuchlichen Trace-Stufen gehören all, debug, entryExit und event.

Im Folgenden sehen Sie ein Beispiel für eine Trace-Zeichenfolge:

```
ObjectGridComponent=level=enabled
```

Sie können Trace-Zeichenfolgen verknüpfen. Verwenden Sie das Symbol `*` (Stern), um einen Platzhalterwert anzugeben, z. B. `ObjectGrid*=all=enabled`. Wenn Sie einen Trace für die IBM Unterstützungsfunktion bereitstellen müssen, ist eine bestimmte Trace-Zeichenfolge erforderlich. So kann beispielsweise die Trace-Zeichenfolge `ObjectGridReplication=debug=enabled` angefordert werden, wenn ein Problem mit der Replikation auftritt.

Trace-Spezifikation

ObjectGrid
Allgemeine Basiscachesteuerkomponente.

ObjectGridCatalogServer
Allgemeiner Katalogservice.

ObjectGridChannel
Statische Kommunikation in der Implementierungstopologie.

ObjectgridCORBA
Dynamische Kommunikation in der Implementierungstopologie.

ObjectGridDataGrid
Die API "AgentManager".

ObjectGridDynaCache
Der dynamische Cacheprovider von WebSphere eXtreme Scale.

ObjectGridEntityManager
Die API ″EntityManager″. Mit der Option ″Projector″ zu verwenden.

ObjectGridEvictors
Integrierte ObjectGrid-Bereinigungsprogramme.

ObjectGridJPA
JPA-Loader (Java Persistence API).

ObjectGridJPACache
JPA-Cache-Plug-ins.

ObjectGridLocking
Sperrenmanager für ObjectGrid-Cacheeinträge.

ObjectGridMBean
Management-Beans.

ObjectGridPlacement
Catalogserverservice für Shard-Verteilung.

ObjectGridQuery
ObjectGrid-Abfrage.

ObjectGridReplication
Replikationsservice.

ObjectGridRouting
Details zum Client/Server-Routing.

ObjectGridSecurity
Sicherheits-Trace.

ObjectGridStats
ObjectGrid-Statistiken.

ObjectGridStreamQuery
Die API ″Stream Query″.

ObjectGridWriteBehind
ObjectGrid-Write-Behind.

Projector
Die Steuerkomponente in der API ″EntityManager″.

QueryEngine
Die Abfragesteuerkomponente für die API ″Object Query″ und die API ″EntityManager Query″.

QueryEnginePlan
Diagnose des Abfrageplans.

Nachrichten

Wenn Sie in einem Protokoll oder in anderen Teilen der Produktschnittstelle eine Nachricht sehen, können Sie die Nachricht mit Hilfe des Komponentenpräfix suchen, um weitere Informationen zu erhalten.

Nachrichten suchen

Wenn Sie eine Nachricht in einem Protokoll finden, kopieren Sie die Nachrichtennummer mit ihrem Buchstabenpräfix und iher Nummer, und suchen Sie im Information Center danach (z. B. `CWOBJ1526I`). Wenn Sie nach der Nachricht suchen, können Sie eine zusätzliche Erläuterung der Nachricht und Beschreibung mögli-

cher Maßnahmen finden, die Sie zum Beheben des Problems verwenden können.

Einen Index der Produktnachrichten finden Sie im Information Center.

Releaseinformationen

Hier finden Sie Links zur Unterstützungswebsite für das Produkt, zur Produktdokumentation und zum letzten Stand der Updates, Einschränkungen und bekannten Problemen für das Produkt.

- „Zugriff auf den letzten Stand von Updates, Einschränkungen und bekannten Problemen"
- „Zugriff auf System- und Softwarevoraussetzungen"
- „Zugriff auf die Produktdokumentation "
- „Zugriff auf die Unterstützungswebsite für das Produkt "
- „Kontaktaufnahme mit der IBM Softwareunterstützung "

Zugriff auf den letzten Stand von Updates, Einschränkungen und bekannten Problemen

Die Releaseinformationen sind auf der Produktunterstützungssite als technische Hinweise verfügbar. Eine Liste aller technischen Hinweise für WebSphere eXtreme Scale finden Sie auf der Unterstützungswebseite.

- Eine Liste der Releaseinformationen für Version 7.0 finden Sie auf der Unterstützungswebseite.
- Eine Liste der Releaseinformationen für Version 6.1 finden Sie auf der Wiki-Site zu den Releaseinformationen.

Zugriff auf System- und Softwarevoraussetzungen

Die Hardware- und Softwarevoraussetzungen finden Sie auf den folgenden Webseiten:

- Detailed system requirements

Zugriff auf die Produktdokumentation

Das vollständige Informationsset finden Sie auf der Bibliotheksseite.

Zugriff auf die Unterstützungswebsite für das Produkt

Wenn Sie nach den neuesten technischen Hinweisen, Downloads und Korrekturen sowie nach Informationen zur Unterstützung suchen, rufen Sie die Unterstützungswebseite auf.

Kontaktaufnahme mit der IBM Softwareunterstützung

Wenn ein Problem mit dem Produkt auftritt, versuchen Sie zuerst, die folgenden Aktionen auszuführen:

- Führen Sie die in der Produktdokumentation beschriebenen Schritte aus.
- Suchen Sie in der Onlinehilfe nach Referenzliteratur.
- Schlagen Sie die Fehlernachrichten in der Nachrichtenreferenz nach.

Sollten Sie das Problem nicht auf diese Weise lösen können, wenden Sie sich an die technische Unterstützung der IBM.

Leistung und bewährte Verfahren

Sie können die Leistung von WebSphere eXtreme Scale verbessern.

WebSphere eXtreme Scale ist zur Verbesserung der Anwendungsleistung konzipiert. Es gibt zahlreiche Faktoren, die sich beim Einsatz von eXtreme Scale auf die Anwendungsleistung auswirken. Dazu gehören unter anderem die Transaktionsgröße, der Kopiermodus, die Serialisierungstechnik und die Partitionsarchitektur. Um die beste Leistung zu erzielen, müssen Sie die Architektur definieren und Ihre Anwendung sorgfältig implementieren.

Bevor Sie versuchen, die Leistung zu verbessern, sollten Sie sich die Informationen in der Dokumentation zu Transaktionen im Handbuch *Produktübersicht* ansehen, um mehr zur Auswahl einer Sperrstrategie für Ihre Umgebung zu erfahren.

Leistung von eXtreme-Scale-Artefakten

Sie können mit verschiedenen Aspekten von eXtreme Scale arbeiten, um die Leistung zu verbessern.

- **Serialisierungsleistung:** Weitere Informationen hierzu finden Sie in der Dokumentation zur Serialisierungsleistung im Handbuch *Produktübersicht*.
- **Transaktionsgröße:** Weitere Informationen hierzu finden Sie in der Dokumentation zu Transaktionsgrößen im Handbuch*Produktübersicht*.
- **Replikationsleistung:** Weitere Informationen hierzu finden Sie in der Dokumentation zur Replikationsleistung im Handbuch *Produktübersicht*.
- **Leistungseinfluss des EntityManager:** Weitere Informationen hierzu finden Sie in der Dokumentation zur Leistung der Schnittstelle ″EntityManager″ im *Programmierhandbuch*.
- **Abfrageoptimierung:** Weitere Informationen hierzu finden Sie in der Dokumentation zur Optimierung von Abfragen zur Leistungsverbesserung im *Programmierhandbuch*.

Bewährte Verfahren

Sie können die Leistung von Maps mit verschiedenen bewährten Verfahren verbessern. Jede Anwendung und jede Umgebung verwendet eine andere Lösung für die Leistungsverbesserung, und eXtreme Scale stellt integrierte Anpassungen für die Leistungsverbesserung zur Verfügung. Sie können die Leistung innerhalb der Anwendungsarchitektur weiterhin verbessern. Verbesserungen werden nur im Kontext der Anwendung und ihrer Architektur implementiert.

- **Bewährte Verfahren für die Sperrleistung:** Sie können zwischen verschiedenen Sperrstrategien wählen, die Einfluss auf die Leistung Ihrer Anwendungen haben. Weitere Informationen finden Sie in der Dokumentation zu den bewährten Verfahren für die Leistung von Sperren im *Programmierhandbuch* .
- **Bewährte Verfahren für die Methode ″CopyMode″:** Sie können zwischen verschiedenen Kopiermodi wählen, mit denen die Art und Weise geändert werden kann, in der eXtreme Scale Einträge verwaltet und kopiert. Weitere Informationen finden Sie in der Dokumentation zu den bewährten Verfahren für die Methode ″CopyMode″ im *Programmierhandbuch* .
- **Bewährte Verfahren für die Schnittstelle ″ObjectTransformer″:** Verwenden Sie die Schnittstelle ″ObjectTransformer″, um Callbacks an die Anwendung zuzulassen, die angepasste Implementierungen gebräuchlicher und kostenintensiver Operationen bereitstellen, wie z. B. Serialisierung und tiefe Kopien von Objekten.

Weitere Informationen finden Sie in der Dokumentation zu den bewährten Verfahren für die Schnittstelle "ObjectTransformer" im *Programmierhandbuch*.

- **Bewährte Verfahrne für Plug-in-Bereinigungsprogramme:** Sie können zwischen den Bereinigungsstrategien LFU (Least Frequently Used) und LRU (Least Recently Used) wählen. Weitere Informationen finden Sie in der Dokumentation zu den bewährten Verfahren für das Plug-in-Bereinigungsprogramms im *Programmierhandbuch* .
- **Bewährte Verfahren für das Standardbereinigungsprogramm:** Eigenschaften für das Standard-TTL-Bereinigungsprogramm (Time-to-Live, Lebensdauer), das das mit jeder BackingMap erstellte Standardbereinigungsprogramm (Evictor) ist. Weitere Informationen finden Sie in der Dokumentation zu den bewährten Verfahren für das Standardbereinigungsprogramm im *Programmierhandbuch*.
- JVM-Optimierung
- „WebSphere Real Time verwenden" auf Seite 222: Dieses Feature ermöglicht eine vorhersehbarere Garbage-Collection mit stabilen, konsistenten Antwortzeiten und Transaktionsdurchsätzen mit eXtreme Scale.

Leistungsüberwachung

eXtreme Scale stellt auch einen Mechanismus für die Leistungsüberwachung bereit. Informationen zu Leistungsüberwachungsstatistiken und -tools finden Sie in den folgenden Abschnitten.

- „PMI-Module" auf Seite 276
- „Leistung mit WebSphere Application Server PMI überwachen" auf Seite 271

Kapitel 11. Glossar

Dieses Glossar enthält Begriffe und Definitionen für WebSphere eXtreme Scale.

In diesem Glossar werden die folgenden Querverweise verwendet:

1. 'Siehe' verweist von einem Terminus zu einem bevorzugt zu verwendenden Synonym oder von einem Akronym oder einer Abkürzung zu der Definition der vollständigen Langform des Terminus.
2. 'Siehe auch' verweist auf einen zugehörigen Terminus oder auf ein Antonym.

Wenn Sie Glossare zu anderen IBM Produkten anzeigen möchten, dann rufen Sie die Website www.ibm.com/software/globalization/terminology auf.

Abfrage.

1. Eine Anforderung von Informationen aus einer Datenbank, die auf bestimmten Bedingungen basiert, z. B. eine Anforderung einer Liste aller Kunden in einer Kundentabelle, deren Kontostand höher als 1000 Euro ist.

2. Eine wiederverwendbare Informationsanforderung über ein oder mehr Modellelemente.

Ableitung. In der objektorientierten Programmierung die Verbesserung oder Erweiterung einer Klasse auf der Basis einer anderen Klasse.

Administrator. Eine Person, die für die Ausführung von Verwaltungstasks wie beispielsweise der Vergabe von Zugriffsberechtigungen und für das Content-Management verantwortlich ist. Administratoren sind auch für die Vergabe von Berechtigungsstufen an Benutzer zuständig.

Agent. Ein Programm, das eine Aktion für einen Benutzer oder ein anderes Programm ohne Benutzereingriff oder nach einem regelmäßigen Zeitplan ausführt und die Ergebnisse zurück an den Benutzer oder das Programm meldet.

Aktualisierbare Sperre. Eine Sperre, die die Absicht aufzeigt, einen Cacheeintrag zu aktualisieren, wenn pessimistisches Sperren verwendet wird.

Aktualisierungspaket. Eine kumulative Sammlung von Programmkorrekturen mit neuen Funktionen. Siehe auch Fixpack, Vorläufiger Fix.

Allgemein. Dieser Begriff bezeichnet das Anzeigen einer Gruppe von Objekten auf abstrakter oder höherer Ebene.

Angepasstes Installationspaket. Ein angepasstes Installationsimage, das mindestens ein Wartungspaket, eine Konfigurationsarchivierungsdatei aus einem eigenständigen Serverprofil, mindestens eine Unternehmensarchivdatei, Scripts und sonstige Dateien enthalten kann, die Unterstützung bei der Anpassung der daraus resultierenden Installation bieten.

Anwendung. Computerprogramme oder Softwarekomponenten, die eine Funktion für die direkte Unterstützung eines oder mehrerer Geschäftsprozesse bereitstellen.

Anwendungsprogrammierschnittstelle (API). Ein Schnittstelle, die einem in einer höheren Programmiersprache geschriebenen Anwendungsprogramm die Verwendung bestimmter Daten oder Funktionen des Betriebssystem oder eines anderen Programms ermöglicht.

Anwendungsserver. Ein Serverprogramm in einem verteilten Netz, das die Ausführungsumgebung für ein Anwendungsprogramm bereitstellt.

APAR. Siehe Authorized Program Analysis Report.

API. Siehe Anwendungsprogrammierschnittstelle (Application Programming Interface).

Arbeitsbereich.

1. Ein Verzeichnis auf Platte, das alle Projektdateien sowie Informationen wie Benutzervorgaben enthält.

2. Ein temporäres Repository für Konfigurationsdaten, das von Clients mit Verwaltungsfunktionen verwendet wird.

3. In Eclipse die Sammlung der Projekte und anderen Ressourcen, die der Benutzer momentan in der Workbench entwickelt. Metadaten zu diesen Ressourcen befinden sich in einem Verzeichnis im Dateisystem. Die Ressourcen können sich in demselben Verzeichnis befinden.

Asynchron. Dieser Begriff bezeichnet Ereignisse, die zeitlich nicht synchronisiert sind oder nicht in regelmäßigen oder voraussagbaren Zeitabständen auftreten.

Asynchrones Messaging. Eine Methode der Kommunikation zwischen Programmen, bei der das Programm eine Nachricht in eine Nachrichtenwarteschlange stellt und dann mit seinen eigenen Anforderungen fortfährt, ohne auf eine Antwort auf die Nachricht zu warten.

Asynchrones Replikat. Ein Shard, das Aktualisierungen nach der Festschreibung der Transaktion empfängt. Diese Methode ist schneller als ein synchrones Replikat, birgt aber das Risiko eines Datenverlusts, weil das asynchrone Replikat aus mehreren Transaktionen hinter dem primären Shard bestehen kann.

Aufruf. Die Aktivierung eines Programms oder einer Prozedur.

Ausdruck. Ein SQL- oder XQuery-Operand oder eine Sammlung von SQL- oder XQuery-Operatoren und -Operanden, die einen einzigen Wert ergibt.

Ausführungs-Trace. Eine Kette von Ereignissen, die in hierarchischem Format auf der Ereignisseite des Integrationstestclients aufgezeichnet und angezeigt wird.

Ausgeschriebener Name. Die Eigenschaft, die den logischen Namen für den Server auf der z/OS-Plattform angibt.

Auslösen. Bei der objektorientierten Programmierung das Veranlassen eines Statusübergangs.

Ausnahme. Eine Bedingung oder ein Ereignis, die bzw. das von einem normalen Prozess nicht verarbeitet werden kann.

Ausnahmebehandlungsroutine. Eine Gruppe von Routinen, die auf eine abnormale Bedingung reagieren. Eine Ausnahmebehandlungsroutine kann die normale Prozessausführung unterbrechen und auch wieder aufnehmen.

Ausstellerzertifikat. Der anerkannte Zertifikateintrag, der in der Regel in einer Truststore-Datei enthalten ist.

Authentifizierter Benutzer. Ein Portalbenutzer, der für die Anmeldung beim Portal einen gültigen Benutzereintrag (Benutzer-ID und Kennwort) verwendet hat. Authentifizierte Benutzer haben Zugriff auf alle öffentlichen Bereiche.

Authentifizierung. Ein Sicherheitsservice, der sicherstellt, dass ein Benutzer eines Computersystems wirklich die Person ist, die er zu sein vorgibt. Allgemeine Mechanismen für die Implementierung dieses Service sind Kennwörter und digitale Signaturen. Die Authentifizierung unterscheidet sich von der Berechtigung. Die Authentifizierung dient nicht zur Erteilung oder Verweigerung von Zugriff auf Systemressourcen.

Authentifizierungsalias. Ein Alias, mit dem die Berechtigung für den Zugriff auf Ressourcenadapter und Datenquellen gewährt wird. Ein Authentifizierungsalias enthält Authentifizierungsdaten einschließlich einer Benutzer-ID und eines Kennworts.

Authorized Program Analysis Report. Eine Anforderung zur Behebung eines Fehlers in einem unterstützten Release eines von IBM gelieferten Programms.

Automatische Erkennung. Die Erkennung von Serviceartefakten in einem Dateisystem, einer externen Registry oder in einer anderen Quelle.

Autonomic Manager. Eine Gruppe von Software- oder Hardwarekomponenten, die über Richtlinien konfiguriert werden und das Verhalten anderer Software- oder Hardwarekomponenten so verwalten, wie es auch ein Benutzer tun würde. Zu einem Autonomic Manager gehört ein Regelkreis, der aus Überwachungs-, Analyse-, Plan- und Ausführungskomponenten besteht.

Autorisierung/Berechtigung. Der Prozess, mit dem einem Benutzer, System oder Prozess entweder uneingeschränkter oder eingeschränkter Zugriff auf ein Objekt, eine Ressource oder eine Funktion gewährt wird.

Bean. Eine Definition oder Instanz einer JavaBeans-Komponente. Siehe auch JavaBeans, Enterprise-Bean.

Bean-Klasse. Bei der EJB-Programmierung (Enterprise JavaBeans) eine Java-Klasse, die eine javax.ejb.EntityBean- oder eine javax.ejb.SessionBean-Klasse implementiert.

Bean-managed Messaging. Über JavaBeans realisiertes Messaging. Eine Funktion des asynchronen Messaging, durch die eine Enterprise-Bean die vollständige Steuerung der Messaging-Infrastruktur erhält.

Bean-Managed Persistence (BMP) . Der Mechanismus, durch den die Datenübertragung zwischen den Variablen einer Entity-Bean und einem Ressourcenmanager von der Entity-Bean verwaltet wird. (Sun)

Bean-Managed Transaction (BMT) . Über JavaBeans realisierte Transaktion. Die Fähigkeit einer Session-Bean, eines Servlet oder einer Anwendungsclientkomponente, eigene Transaktionen direkt ohne Unterstützung eines Containers zu verwalten.

Bean Scripting Framework. Eine Architektur, mit der die Funktionen von Scripting-Sprachen in Java-Anwendungen eingebunden werden können.

Befehlszeile. Eine leere Zeile in der Anzeige, in der Befehle, Optionsnummern oder ausgewählte Optionen eingegeben werden können.

Berechtigung. Die Berechtigung zur Ausführung von Aktivitäten wie Lesen und Schreiben von lokalen Dateien, Erstellen von Netzverbindungen und Laden des nativen Codes.

Berechtigungsnachweis. In der JAAS-Architektur (Java Authentication and Authorization Service) eine Subjektklasse mit sicherheitsrelevanten Attributen. Diese Attribute können Informationen enthalten, die für die Authentifizierung des Subjekts für neue Services verwendet wird.

Berechtigungsrichtlinie. Eine Richtlinie, deren Richtlinienziel ein Geschäftsservice ist und deren Vertrag mindestens eine Zusicherung enthält, auf deren Basis die Berechtigung zur Ausführung einer Kanalaktion erteilt wird.

Berechtigungstabelle. Eine Tabelle, die die Informationen für die Zuordnung einer Rolle zu einem Benutzer oder zu einer Gruppe enthält, die den zulässigen Zugriff eines Clients auf eine bestimmte Ressource definieren.

Bereinigungsprogramm. Eine Komponente, die die Zugehörigkeit von Einträgen in jeder BackingMap-Instanz steuert. Teilcaches können Bereinigungsprogramme verwenden, um Daten ohne Beeinträchtigung der Datenbank automatisch aus dem Cache zu entfernen.

Bibliothek.

1. Eine Gruppe von Modellelementen einschließlich der zugehörigen Geschäftselemente, Prozesse, Tasks, Ressourcen und Organisationen.

2. Ein Projekt, das für Entwicklung, Versionsmanagement und Organisation gemeinsam genutzter Ressourcen verwendet wird. In einer Bibliothek kann nur ein Teil der Artefakttypen erstellt und gespeichert werden, z. B. Geschäftsobjekte und Schnittstellen.

Binärformat. Die Darstellung eines Dezimalwerts, bei der jedes Feld zwei oder vier Byte lang sein muss. Das Vorzeichen (+ oder -) wird im Feld in dem Bit ganz links angegeben, die restlichen Bit des Felds geben den Zahlenwert an. Positive Zahlen haben im Bit für das Vorzeichen eine 0 und werden im Standardformat (″True Format″) angegeben. Negative Zahlen haben im Bit für das Vorzeichen eine 1 und werden im Zweierkomplementformat (″Two's Complement Format″) angegeben.

Bindung im Zellenbereich. Ein Bindungsbereich, in dem die Bindung nicht spezifisch und nicht einem Knoten oder Server zugeordnet ist. Diese Art Namensbindung wird unter dem persistenten Ausgangskontext einer Zelle erstellt.

BMP. Siehe Bean-managed Persistence.

BMT. Siehe Bean-managed Transaction.

Boot-Programm. Ein kleines Programm, mit dem umfangreichere Programme während der Systeminitialisierung geladen werden können.

Bootstrapping. Der Prozess, bei dem eine Ausgangsreferenz des Namensservice abgerufen wird. Die Bootstrap-Einstellung und der Hostname bilden den Ausgangskontext für JNDI-Referenzen (Java Naming and Directory Interface).

Bottom-up-Entwicklung. Bei Web-Services der Prozess der Entwicklung eines Service auf der Basis eines vorhandenen Artefakts wie z. B. einer Java-Bean oder einer Enterprise-Bean und nicht auf der Basis einer WSDL-Datei (WSDL = Web Services Description Language).

Bytecode. Ein systemunabhängiger Code, der vom Java-Compiler generiert und vom Java-Interpreter ausgeführt wird. (Sun)

Cacheinstanzressource. Eine Position, unter der jede Java EE-Anwendung (Java EE = Java Platform Enterprise Edition) Daten speichern, verteilen und gemeinsam nutzen kann.

Cachereplikation. Die gemeinsame Nutzung von Cache-IDs, Cacheeinträgen und Cacheinvalidierungen mit anderen Servern innerhalb einer Replikationsdomäne.

CIP. Siehe Angepasstes Installationspaket (Customized Installation Package).

Client. Ein Softwareprogramm oder ein Computer, der Services von einem Server anfordert. Siehe auch Host.

Clientanwendung. Eine Anwendung, die auf einer Workstation ausgeführt wird und über eine Verbindung zu einem Client verfügt, über die die Anwendung auf die Warteschlangensteuerungsservices auf einem Server zugreifen kann.

Client/Server. Bezeichnung des Interaktionsmodells bei der verteilten Datenverarbeitung, in dem ein Programm auf einem Computer eine Anforderung an ein Programm auf einem anderen Computer sendet und die Antwort abwartet. Das anfordernde Programm ist ein Client und das antwortende Programm ein Server.

Cloudscape. Ein integrierbares, vollständig Java-gestütztes, objektrelationales Datenbankverwaltungssystem (ORDBMS, Object-Relational Database Management System).

Cluster. Eine Gruppe von Anwendungsservern, die zusammenarbeitet, um Funktionen wie Lastausgleich und Ausweichbetrieb bereitzustellen.

Command-Bean. Ein Proxy, der eine einzelne Operation mithilfe der Methode execute() aufrufen kann.

Containerserver. Eine Serverinstanz, die mehrere Shards haben kann. Eine Java Virtual Machine (JVM) kann mehrere Containerserver haben.

Converter. Bei der EJB-Programmierung (EJB = Enterprise JavaBeans) eine Klasse, die eine Datenbankdarstellung in einen Objekttyp umsetzt (und umgekehrt).

Create-Methode. In Enterprise-Beans eine Methode, die in der Home-Schnittstelle definiert ist und von einem Client aufgerufen wird, um eine Enterprise-Bean zu erstellen. (Sun)

CSV-Datei. Eine Datei, deren Datensätze durch Kommas voneinander getrennte Felder enthalten.

Dämon. Ein Programm, das unbeaufsichtigt ausgeführt wird und fortlaufend oder in regelmäßigen Abständen Funktionen wie die Netzsteuerung ausführt.

Dashboard. Eine Webseite, die einen oder auch mehrere Viewer umfassen kann, mit deren Hilfe Geschäftsdaten grafisch dargestellt werden können.

Daten-Grid. Ein System für den Zugriff auf Terabytes oder Petrabytes von Daten.

DB2. Eine Familie von IBM Lizenzprogrammen für die Verwaltung relationaler Datenbanken.

Deadlock. Eine Bedingung, bei der zwei unabhängige Steuerungsthreads blockiert werden, wobei jeder vom anderen eine Aktion erwartet. Deadlocks treten häufig dann auf, wenn Synchronisationsmechanismen zur Vermeidung von Konkurrenzsituationen hinzugefügt werden.

Demilitarized Zone (DMZ). Eine Konfiguration, die mehrere Firewalls enthält, um zwischen ein Unternehmens-Intranet und ein öffentliches Netz wie das Internet mehrere Sicherheitsebenen hinzuzufügen.

Deployment Manager. Ein Server, der den Betrieb einer logischen Gruppe oder einer Zelle anderer Server verwaltet.

Destination. Ein Exitpunkt, der verwendet wird, um Dokumente an ein Back-End-System oder einen Handelspartner zu liefern.

Differenziert. Dieser Begriff bezeichnet das Anzeigen der Details zu einem einzelnen Objekt.

Digitales Zertifikat. Ein elektronisches Dokument, das zur Identifikation einer Person, eines Systems, eines Servers, eines Unternehmens oder einer anderen Entität sowie zur Zuordnung eines öffentlichen Schlüssels zu der Entität dient. Ein digitales Zertifikat wird von einer Zertifizierungsstelle ausgestellt und von dieser Instanz digital signiert.

DMZ . Siehe Demilitarized Zone.

DNS. Siehe Domain Name System.

Document Type Definition (DTD). Die Regeln, die die Struktur einer bestimmten Klasse von SGML- oder XML-Dokumenten festlegen. Die DTD definiert die Struktur mit Elementen, Attributen und Notationen und richtet Integritätsbedingungen dafür ein, wie die einzelnen Elemente, Attribute und Notationen in der jeweiligen Klasse von Dokumenten verwendet werden können.

Domain Name System (DNS). Das verteilte Datenbanksystem, das Domänennamen IP-Adressen zuordnet.

Domäne. Ein Objekt, ein Symbol oder ein Container, der andere Objekte enthält, die die Ressourcen einer Domäne darstellen. Die Objektdomäne kann für die Verwaltung dieser Ressourcen verwendet werden.

Do-while-Schleife. Eine Schleife, die die gleiche Aktivitätenfolge so lange wiederholt, bis eine bestimmte Bedingung erfüllt ist. Anders als eine While-Schleife testet eine Do-while-Schleife ihre Bedingung am Ende der Schleife. Dies bedeutet, dass ihre Aktivitätenfolge immer mindestens einmal ausgeführt wird.

Dropdown. Siehe Pulldown.

DTD. Siehe Dokumenttypdefinition.

DTD-Dokumentdefinition. Eine Beschreibung oder ein Layout eines XML-Dokuments, die bzw. das auf einer XML-Dokumenttypdefinition basiert.

Durchsatz. Die Kennzahl für das Arbeitsvolumen, das von einem Gerät, wie einem Computer oder Drucker, im Verlauf eines bestimmten Zeitraums ausgeführt wird, beispielsweise die Anzahl Jobs pro Tag.

Dynamischer Cache. Eine Konsolidierung mehrerer Caching-Aktivitäten einschließlich der entsprechenden Servlets, Web-Services und WebSphere-Befehle in einem Service, in dem diese Aktivitäten Konfigurationsparameter gemeinsam nutzen und zusammenwirken, um die Leistung zu verbessern.

Dynamischer Cluster. Ein Servercluster, der basierend auf den von den Cluster-Membern erfassten Leistungsdaten anhand von Wertigkeiten die Arbeitslast dynamisch auf seine Cluster-Member verteilt.

EAR . Siehe Unternehmensarchiv.

EAR-Projekt. Siehe Unternehmensanwendungsprojekt.

Eclipse. Eine Open-Source-Initiative, die unabhängigen Softwareanbietern (ISVs) und anderen Toolentwicklern eine Standardplattform für die Entwicklung von plug-kompatiblen Anwendungsentwicklungstools zur Verfügung stellt.

Edition. Eine Nachfolgegeneration der Implementierung einer bestimmten Gruppe versionsgesteuerter Artefakte.

Editorbereich. In Eclipse und bei Eclipse-basierten Produkten der Bereich des Workbench-Fensters, in dem Dateien zum Bearbeiten geöffnet werden.

Eigenschaft. Ein Merkmal eines Objekts, durch das das Objekt beschrieben wird. Eine Eigenschaft kann geändert werden. Eigenschaften können unter anderem Namen, Typ, Wert oder Verhalten eines Objekts beschreiben.

Eigenständig. Unabhängig von allen anderen Einheiten, Programmen oder Systemen. In einer Netzumgebung greift ein eigenständiges System auf alle erforderlichen Ressourcen lokal zu.

Eigenständiger Server. Ein Katalogservice oder Containerserver, der vom Betriebssystem verwaltet wird, das den Serverprozess startet und stoppt.

Eingabeaufforderung. Eine Komponente einer Aktion, mit der angegeben wird, dass eine Benutzereingabe für ein Feld erforderlich ist, bevor ein Übergang zur Ausgabeanzeige erfolgt.

Eingangsunterbrechungspunkt. Ein Unterbrechungspunkt, der für ein Komponentenelement gesetzt ist und erreicht wird, bevor das Komponentenelement aufgerufen wird.

EJB. Siehe Enterprise JavaBeans.

EJB-Abfrage. In der EJB-Abfragesprache eine Zeichenfolge, die Folgendes enthält: eine optionale SELECT-Klausel, die die EJB-Objekte angibt, die zurückgegeben werden sollen; eine FROM-Klausel, die die Bean-Collections benennt; eine optionale WHERE-Klausel, die Suchvergleichselemente für die Collections enthält; eine optionale ORDER BY-Klausel, die die Sortierung der Ergebniscollection angibt; und Eingabeparameter, die den Argumenten der Finder-Methode entsprechen.

EJB-Container. Ein Container, der den EJB-Komponentenvertrag der Java EE-Architektur implementiert. In diesem Vertrag wird eine Laufzeitumgebung für Enterprise-Beans angegeben, die Services für die Sicherheit, den gemeinsamen Zugriff, das Lebenszyklusmanagement, Transaktionen, die Implementierung und anderes enthält. (Sun)

EJB-Factory. Eine Access-Bean, die die Erstellung einer Enterprise-Bean-Instanz oder die Suche nach einer solchen Instanz vereinfacht.

EJB-Home-Objekt. Bei der EJB-Programmierung (Enterprise JavaBeans) ein Objekt, das die Lebenszyklusoperationen (Erstellen, Entfernen, Suchen) für eine Enterprise-Bean bereitstellt. (Sun)

EJB-JAR-Datei. Ein Java-Archiv, das ein EJB-Modul enthält. (Sun)

EJB-Kontext. In Enterprise-Beans ein Objekt, das einer Enterprise-Bean ermöglicht, vom Container bereitgestellte Services aufzurufen und Informationen über den Aufrufenden einer Clientmethode abzurufen. (Sun)

EJB-Modul. Eine Softwareeinheit, die sich aus einer oder mehreren Enterprise-Beans und einem EJB-Deployment-Deskriptor besteht. (Sun)

EJB-Objekt. Bei Enterprise-Beans ein Objekt, dessen Klasse die ferne Schnittstelle der Enterprise-Bean implementiert (Sun).

EJB-Projekt. Ein Projekt, das die Ressourcen enthält, die für EJB-Anwendungen benötigt werden. Hierzu zählen Enterprise-Beans, Home-Schnittstellen, Local- und Remote-Schnittstellen, JSP-Dateien, Servlets und Implementierungsdeskriptoren.

EJB-Referenz. Ein logischer Name, der von einer Anwendung zur Lokalisierung der Home-Schnittstelle einer Enterprise-Bean in der Zielbetriebsumgebung verwendet wird.

EJB-Server. Software, die Services für einen EJB-Container bereitstellt. Ein EJB-Server kann einen oder mehrere EJB-Container aufnehmen. (Sun)

EJB-Vererbung. Eine Form der Vererbung, bei der eine Enterprise-Bean die Eigenschaften, Methoden und Deskriptorattribute für die Steuerung auf Methodenebene von einer anderen Enterprise-Bean in derselben Gruppe erbt.

Endpunkt.

1. Eine JCA-Anwendung oder ein anderer Clientkonsument eines Ereignisses aus dem unternehmensweiten Informationssystem.

2. Das System, das den Ursprung oder das Ziel einer Sitzung darstellt.

Endpunkt-Listener. Der Punkt oder die Adresse, an dem bzw. der eingehende Nachrichten für einen Web-Service von einem Service Integration Bus empfangen werden.

Engpass. Eine Situation im System, die eintritt, wenn die Konkurrenz um eine Ressource sich auf die Leistung auswirkt.

Enterprise-Bean. Eine Komponente, die eine Geschäfts-Task oder Geschäftsentität implementiert und in einem EJB-Container enthalten ist. Entity-Beans, Session-Beans und Message-Driven Beans sind verschiedene Typen von Enterprise-Beans. (Sun) Siehe auch Bean.

Enterprise JavaBeans (EJB). Eine von Sun Microsystems definierte Komponentenarchitektur für die Entwicklung und Implementierung objektorientierter, verteilter Unternehmensanwendungen (Java EE).

Enterprise Service Bus (ESB). Eine flexible Konnektivitätsinfrastruktur für die Integration von Anwendungen und Services, die eine flexible und einfach zu verwaltende Strategie für die SOA-Implementierung (SOA = service-orientierte Architektur) bietet.

Entität.

1. Eine einfache Java-Klasse, die eine Zeile in einer Datenbanktabelle oder einen Eintrag in einer Zuordnung darstellt.

2. In Markup-Sprachen wie XML eine Sammlung von Zeichen, die als Einheit referenziert werden können, beispielsweise für die Einbindung von häufig wiederholtem Text oder Sonderzeichen in einem Dokument.

Entity-Bean. In der EJB-Programmierung eine Enterprise-Bean, die persistente Daten darstellt, die in einer Datenbank verwaltet werden. Jede Entity-Bean hat eine eigene Entität. (Sun)

Entserialisierung. Eine Methode für die Konvertierung einer serialisierten Variablen in Objektdaten.

Ereignis.

1. Die Änderung eines Status wie beispielsweise der Abschluss oder das Fehlschlagen einer Operation, eines Geschäftsprozesses oder einer Benutzertask, die eine nachfolgende Aktion wie beispielsweise das Speichern der Ereignisdaten in einem Datenrepository oder das Aufrufen eines anderen Geschäftsprozesses auslösen kann.

2. Eine Änderung an den in einem EIS (Enterprise Information System) gespeicherten Daten, die vom Adapter verarbeitet und zur Lieferung von Geschäftsobjekten vom EIS zu den Endpunkten (Anwendungen) verwendet wird, die über die Änderung benachrichtigt werden müssen.

Erstellungsdefinitionsdatei. Eine XML-Datei, die Komponenten und Merkmale für ein angepasstes Installationspaket (CIP, Customized Installation Package) angibt.

Erstellungspfad. Der Pfad, der während der Kompilierung des Java-Quellcodes verwendet wird, um referenzierte Klassen zu finden, die sich in anderen Projekten befinden.

Erstellungsplan. Eine XML-Datei, die die Verarbeitung definiert, die zum Erstellen der Generierungsausgabe erforderlich ist und die das System angibt, auf dem die Verarbeitung stattfindet.

Erstellungszeitdaten. Objekte, die vom Umsetzungsprogramm nicht verwendet werden, wie beispielsweise EDI-Standards, ROD-Dokumenttypen (ROD = Record Oriented Data; satzorientierte Daten) und Zuordnungen.

ESB. Siehe Enterprise Service Bus (ESB).

Exklusive Sperre. Eine Sperre, die verhindert, dass Anwendungsprozesse ausgeführt werden, die gleichzeitig auf Datenbankdaten zugreifen. Siehe auch Gemeinsame Sperre.

Export. Eine offene Schnittstelle eines SCA-Moduls (SCA = Service Component Architecture), die einen Geschäftsservice für externe Systeme und Benutzer bereitstellt. Ein Export verfügt über eine Bindung, die definiert, wie auf den Service von Serviceanforderern zugegriffen werden kann, z. B. als Web-Service.

Exportdatei.

1. Eine Datei, die während des Entwicklungsprozesses für eingehende Operationen erstellt wurde und die Konfigurationseinstellungen für die Eingangsverarbeitung enthält.

2. Eine Datei, die exportierte Daten enthält.

Extensible Markup Language (XML). Eine Standardmetasprache für die Definition von Markup-Sprachen, die auf Standard Generalized Markup Language (SGML) basiert.

eXtreme-Scale-Grid. Ein Datenmuster, das für die Interaktion mit eXtreme Scale verwendet wird, wenn sich alle Daten und Clients in einem einzigen Prozess befinden.

Factory. In der objektorientierten Programmierung eine Klasse, die zum Erstellen von Instanzen einer anderen Klasse verwendet wird. Eine Factory wird verwendet, um die Erstellung von Objekten einer bestimmten Klasse auf einen Bereich zu isolieren, damit neue Funktionen ohne weitreichende Codeänderungen bereitgestellt werden können.

Failover. Eine automatische Operation, mit der auf ein redundantes oder Bereitschaftssystem umgeschaltet werden kann, wenn eine Software-, Hardware- oder Netzunterbrechung eintritt.

Fehler. Eine Abweichung zwischen einem berechneten, festgestellten oder gemessenen Wert oder einer solchen Bedingung und dem wahren, definierten oder theoretisch richtigen Wert bzw. der entsprechenden Bedingung.

Fehlerhafte Leseoperation. Eine Leseanforderung, die keinen Sperrmechanismus besitzt. Das bedeutet, dass Daten gelesen werden können, die später zurückgesetzt werden, was zu einer Inkonsistenz zwischen den gelesenen Daten und den Daten in der Datenbank führt.

Fehlerprotokollstrom. Ein fortlaufender Fluss von Fehlerinformationen, die in einem vordefinierten Format übertragen werden.

Firewall . Eine Netzkonfiguration, die in der Regel Hardware und Software umfasst und verhindert, dass nicht autorisierter Datenverkehr in ein sicheres Netz eingeht bzw. dieses verlässt.

Fixpack. Eine kumulative Gruppe von Fixes (Programmkorrekturen), die zwischen den geplanten Refresh-Packs, Produktaktualisierungen oder Releases bereitgestellt wird. Fixpacks ermöglichen es den Kunden, ihre Systeme auf eine bestimmte Wartungsstufe umzustellen. Siehe auch Vorläufiger Fix.

For-Schleife. Eine Schleife, die eine angegebene Anzahl von Wiederholungen für eine bestimmte Aktivitätenfolge ausführt.

Garbage-Collection. Eine Routine, die den Hauptspeicher durchsucht, um Speicher aus Programmsegmenten oder inaktiven Daten zurückzufordern.

Gehäuse. Ein Metallrahmen, in den verschiedene elektronische Komponenten eingebaut werden.

Geltungsbereich.

1. Eine Spezifikation der Begrenzung, innerhalb deren Systemressourcen verwendet werden können.

2. Bei Web-Services eine Eigenschaft, die die Lebensdauer des Objekts angibt, das die Aufrufanforderung bearbeitet.

Gemeinsame Sperre. Eine Sperre, die gleichzeitig aktive Anwendungsprozesse auf Leseoperationen für Datenbankdaten beschränkt.

General Inter-ORB Protocol (GIOP). Ein Protokoll, das von der Common Object Request Broker Architecture (CORBA) zum Festlegen des Nachrichtenformats verwendet wird.

Generisches Objekt. Ein Objekt, mit dem in API-Aufrufen und XPATH-Ausdrücken auf Konzepte, angepasste Entitäten oder Sammlungen verwiesen wird. Beispiel: Mit dem XPATH-Ausdruck '/WSRR/GenericObject' werden alle Konzepte aus WebSphere Service Registry and Repository abgerufen.

Gerüst. Der Entwurf für eine Implementierungsklasse.

Getter-Methode. Eine Methode, die den Zweck hat, den Wert einer Instanz oder einer Klassenvariablen abzurufen. Dadurch kann ein anderes Objekt den Wert einer seiner Variablen herausfinden.

GIOP . Siehe General Inter-ORB Protocol.

Global.

1. Dieser Begriff bezeichnet die Eigenschaft eines Elements, das für alle Prozesse innerhalb eines Arbeitsbereichs bereitgestellt wird. Ein globales Element wird im Projektbaum aufgeführt und kann in mehreren Prozessen verwendet werden. Tasks, Prozesse, Repositorys und Services können entweder global oder lokal sein. Global bedeutet hier, dass von allen Prozessen eines Projekts auf sie verwiesen werden kann, lokal bedeutet hingegen, dass sie einem bestimmten Prozess zugeordnet sind.

2. Dieser Begriff bezeichnet Informationen, die für mehr als ein Programm oder eine Unterroutine verfügbar sind.

Globale Instanz-ID. Eine global eindeutige ID, die entweder von der Anwendung oder vom Emitter generiert und als Primärschlüssel für die Ereignisidentifikation verwendet wird.

Globales Attribut. In XML ein Attribut, das als untergeordnetes Schemaelement und nicht als Teil einer komplexen Typdefinition deklariert ist. Auf globale Attribute kann unter Verwendung des Attributs ref in einem oder auch in mehreren Inhaltsmodellen verwiesen werden.

Globales Element. In XML ein Element, das als untergeordnetes Schemaelement und nicht als Teil einer komplexen Typdefinition deklariert ist. Auf globale Elemente kann unter Verwendung des Attributs ref in einem oder auch in mehreren Inhaltsmodellen verwiesen werden.

Globale Sicherheit. Die globale Sicherheit bezieht sich auf alle Anwendungen, die in der Umgebung ausgeführt werden, und legt fest, ob Sicherheitseinrichtungen zum Einsatz kommen. Ferner legt sie den Typ des für die Authentifizierung verwendeten Registry fest sowie andere Werte, von denen viele Standardeinstellungen sind.

Globale Transaktion. Eine wiederherstellbare Arbeitseinheit, die durch einen oder mehrere Ressourcenmanager in einer Umgebung für verteilte Transaktionen ausgeführt und von einem externen Transaktionsmanager koordiniert wird.

Globale Variable. Eine Variable, die verwendet wird, um die ihr während der Umsetzung zugeordneten Werte aufzunehmen und zu bearbeiten. Sie wird von verschiedenen Zuordnungen und Dokumentumsetzungen gemeinsam verwendet. Einer von drei Variablentypen, die von der Zuordnungsbefehlssprache von Data Interchange Services unterstützt werden.

Gruppe.

1. Ein Benutzerverbund, der Zugriffsberechtigungen für geschützte Ressourcen gemeinsam benutzen kann.

2. Ein Satz zusammengehöriger Dokumente in einem Austausch. Ein Austausch muss keine Gruppen enthalten, kann aber auch viele Gruppen enthalten.

3. Zwei oder mehrere Personen in einem Bereich, die für die Zugehörigkeit zu einem Bereich in einer Gruppe zusammengefasst werden.

HA. Siehe Hohe Verfügbarkeit (HA, High Availability).

HA-Gruppe. Eine Sammlung mit mindestens einem Mitglied, die zur Bereitstellung einer hohen Verfügbarkeit für einen Prozess genutzt wird.

HA-Richtlinie. Eine Gruppe von Regeln, die für eine HA-Gruppe definiert sind. Diese Regelgruppe gibt vor, ob null (0) oder mehr Mitglieder aktiviert sind. Die Richtlinie wird einer bestimmten HA-Gruppe zugeordnet, indem die Übereinstimmungskriterien der Richtlinie mit dem Gruppennamen abgeglichen werden.

High Availability Manager. Ein Gerüst, in dem die Stammgruppenzugehörigkeit bestimmt und Statusinformationen zwischen den Stammgruppen-Membern übertragen wird.

Hohe Verfügbarkeit (HA, High Availability). Bezeichnung für ein Clustersystem, das neu konfiguriert wird, wenn ein Knoten- oder Dämonfehler auftritt, damit die Workloads auf die verbleibenden Knoten im Cluster umverteilt werden können.

Host.

1. Ein Computer, der mit einem Netz verbunden ist und einen Zugriffspunkt auf dieses Netz bereitstellt. Der Host kann ein Client, ein Server oder Client und Server gleichzeitig sein.

2. Bei der Leistungsprofilerstellung ein System, das über Prozesse verfügt, für die ein Profil erstellt werden soll. Siehe auch Server.

Hostname.

1. Bei der Internetkommunikation der Name eines Computers. Der Hostname kann ein vollständig qualifizierter Domänenname wie mycomputer.city.company.com oder ein bestimmter Teilname wie mycomputer sein.

2. Der Netzname für einen Netzadapter auf einer physischen Maschine, auf der der Knoten installiert ist.

Hostsystem. Ein Großrechnersystem innerhalb eines Unternehmens, das als Host für 3270-Anwendungen eingesetzt wird. Bei Entwicklungstools für 3270-Terminal-Services verwendet der Entwickler den 3270-Terminal-Service-Recorder, um eine Verbindung zum Hostsystem herzustellen.

HTTP over SSL (HTTPS). Ein Web-Protokoll für sichere Transaktionen, das Seitenanforderungen von Benutzern und vom Web-Server zurückgegebene Seiten verschlüsselt und entschlüsselt.

HTTPS.

1. Siehe HTTP over SSL.

2. Siehe Hypertext Transfer Protocol Secure.

Hypertext Transfer Protocol Secure (HTTPS). Ein Internet-Protokoll, das von Webservern und Webbrowsern für die sichere Übertragung und Anzeige von Hypermedia-Dokumenten im Internet verwendet wird.

IDE . Siehe Integrierte Entwicklungsumgebung (Integrated Development Environment).

If-then-Regel. Eine Regel, in der die Aktion ('then'-Teil) nur dann ausgeführt wird, wenn die Bedingung ('if'-Teil) zutrifft.

IIOP. Siehe Internet Inter-ORB Protocol.

Implementieren. Dateien oder Software in einer Betriebsumgebung installieren. In Java EE (Java Platform Enterprise Edition) umfasst das Implementieren die Erstellung eines Implementierungsdeskriptors, der für den zu implementierenden Anwendungstyp geeignet ist.

Implementierphase. Siehe Implementierungsphase.

Implementierungscode. Zusätzlicher Code, der den Bean-Implementierungscode aktiviert, der von einem Anwendungsentwickler zur Verwendung in einer bestimmten EJB-Laufzeitumgebung geschrieben wurde. Implementierungscode kann mithilfe von Tools generiert werden, die der Anbieter des Anwendungsservers liefert.

Implementierungsdeskriptor. Eine XML-Datei (Extensible Markup Language), die den Einsatz eines Moduls oder einer Anwendung beschreibt, indem Sie Konfigurations- und Containeroptionen festlegt. Ein EJB-Deployment-Deskriptor übergibt beispielsweise Informationen zur Verwaltung und Steuerung einer Enterprise-Bean an einen EJB-Container.

Implementierungsphase. Eine Phase, in der Operationen zur Erstellung der Hosting-Umgebung für Anwendungen und zur Implementierung dieser Anwendungen ausgeführt werden. In dieser Phase werden auch die Ressourcenabhängigkeiten der Anwendung sowie Betriebsbedingungen, Kapazitätsanforderungen, Integritätsbedingungen und Zugriffsbeschränkungen aufgelöst.

Implementierungsrichtlinie. Eine optionale Methode für die Konfiguration einer eXtreme-Scale-Umgebung auf der Basis verschiedener Elemente, wie z. B. Anzahl der Systeme, Server, Partitionen, Replikate (einschließlich Replikattyp) und Heap-Speichergrößen für jeden Server.

Implementierungstopologie. Die Konfiguration von Servern und Clustern in einer Implementierungsumgebung und die physischen und logischen Beziehungen zwischen diesen Einheiten.

Implementierungsumgebung. Eine Sammlung konfigurierter Cluster, Server und Middlewarekomponenten, die zusammenarbeiten, um eine Umgebung bereitzustellen, in der Softwaremodule unterstützt werden können. Eine Implementierungsumgebung kann beispielsweise einen Host für Nachrichtenziele, einen Prozessor oder eine Sortierkomponente für Geschäftsereignisse und Verwaltungsprogramme umfassen.

Implementierungsverzeichnis. Das Verzeichnis, in dem sich die veröffentlichte Serverkonfiguration und die Webanwendung auf dem System, auf dem der Anwendungsserver installiert ist, befinden.

Import.

1. Das Entwicklungsartefakt, das zum Importieren eines Service verwendet wird, der nicht in einem Modul integriert ist.

2. Der Punkt, über den ein SCA-Modul auf einen externen Service (d. h. einen Service außerhalb des SCA-Moduls) in derselben Weise zugreift, wie dies bei einem lokalen Service möglich ist. Ein Import definiert die Interaktion zwischen dem SCA-Modul und dem Serviceanbieter. Ein Import verfügt über eine Bindung sowie mindestens eine Schnittstelle.

Index. Eine Gruppe von Zeigern, die logisch nach den Werten eines Schlüssels sortiert sind. Indizes ermöglichen den schnellen Zugriff auf Daten und können die Eindeutigkeit der SChlüsselwerte für die Zeilen in der Tabelle erzwingen.

Information Center. Eine Sammlung von Informationen, die Benutzern eines oder mehrerer Produkte zur Unterstützung bereitgestellt wird, die vom Produkt gesondert gestartet werden kann und die eine Liste der Abschnitte zur Navigation sowie eine Suchmaschine enthält.

Installationspaket. Eine installierbare Einheit eines Softwareprodukts. Softwareproduktpakete sind separat installierbare Einheiten, die unabhängig von anderen Paketen dieses Softwareprodukts eingesetzt werden können.

Installationsziel. Das System, auf dem ausgewählte Installationspakete installiert werden.

Instanz. Ein spezielles Vorkommen eines Objekts, das einer Klasse angehört.

Instanzieren. Das Darstellen einer Abstraktion durch eine konkrete Instanz.

Integrated Development Environment (IDE) . Eine Gruppe von Softwareentwicklungstools wie Quelleneditoren, Compiler und Debugger, auf die über eine gemeinsame Benutzerschnittstelle zugegriffen werden kann.

Integrierter Server. Ein Katalogservice oder Containerserver, der sich in einem vorhandenen Prozess befindet und innerhalb des Prozesses gestartet und gestoppt wird.

Internet Inter-ORB Protocol (IIOP). Ein Protokoll, das für die Kommunikation zwischen CORBA-Object-Request-Brokern (CORBA = Common Object Request Broker Architecture) eingesetzt wird.

Internet Protocol (IP). Ein Protokoll, das Daten über ein Netz oder über miteinander verbundene Netze leitet. Dieses Protokoll agiert als Mittler zwischen den höheren Protokollschichten und dem physischen Netz.

IP. Siehe Internet Protocol.

IP-Sprayer. Eine Einheit, die sich zwischen den von den Benutzern eingehenden Anforderungen und den Anwendungsserverknoten befindet und Anforderungen an Knoten weiterleitet.

Iteration. Siehe Schleife.

Iterator. Eine Klasse oder Anweisung, die verwendet wird, um die Objekte einer Objektgruppe nacheinander zu durchlaufen.

JAAS. Siehe Java Authentication and Authorization Service.

JAF . Siehe JavaBeans Activation Framework.

JAR-Datei. Eine Java-Archivdatei. Siehe auch Webarchiv, Unternehmensarchiv.

Java. Eine objektorientierte Programmiersprache für portierbaren, interpretierenden Code, die die Interaktion zwischen fernen Objekten unterstützt. Java wurde von Sun Microsystems, Incorporated entwickelt und spezifiziert.

Java API für XML (JAX). Eine Gruppe Java-basierter Anwendungsprogrammierschnittstellen für die Verarbeitung verschiedener Operationen, an denen über Extensible Markup Language (XML) definierte Daten beteiligt sind.

Java-Archiv. Ein Format für komprimierte Dateien, mit dem alle Ressourcen, die zur Installation und Ausführung eines Java-Programms erforderlich sind, in einer einzigen Datei gespeichert werden können. Siehe auch Webarchiv, Unternehmensarchiv.

Java Authentication and Authorization Service (JAAS). In Java EE technology, a standard API for performing security-based operations. Über JAAS können Services die Authentifizierung und Berechtigung von Benutzern ausführen. Gleichzeitig bleiben die zugehörigen Anwendungen unabhängig von den zugrunde liegenden Technologien.

JavaBeans . Ein für Java von Sun Microsystems definiertes, portierbares und plattformunabhängiges Komponentenmodell, das wiederverwendbar ist. Siehe auch Bean.

JavaBeans Activation Framework (JAF) . Eine Standarderweiterung für die Java-Plattform, die beliebige Datentypen und verfügbare Operationen bestimmt und eine Bean so instanzieren kann, dass sie relevante Services ausführt.

Java Command Language. Eine Scripting-Sprache für die Java-Umgebung, die zur Erstellung von Webinhalten und zur Steuerung von Java-Anwendungen benutzt werden kann.

Java Connector Security. Eine Architektur, die zur Erweiterung des End-to-End-Sicherheitsmodells für Java EE-basierte Anwendungen entworfen wurde, so dass dieses auch unternehmensweite Informationssysteme (EIS = Enterprise Information Systems) umfassen kann.

Java Database Connectivity (JDBC) . Ein Industriestandard für datenbankunabhängige Konnektivität zwischen der Java-Plattform und verschiedenen Datenbanken. Die JDBC-Schnittstelle stellt eine Schnittstelle auf Aufrufebene für den SQL-basierten und den XQuery-basierten Datenbankzugriff bereit.

Java-Datei. Eine Quellendatei (mit der Erweiterung .java), die bearbeitet und in Bytecode (eine Datei mit der Erweiterung .class) kompiliert werden kann.

Javadoc.

1. Ein Tool, das die Deklarationen und Dokumentationskommentare in einer Gruppe von Quellendateien syntaktisch analysiert und eine Reihe von HTML-Seiten erstellt, die die Klassen, untergeordneten Klassen, Schnittstellen, Konstruktoren, Methoden und Felder beschreiben. (Sun)

2. Dieser Begriff bezeichnet ein Tool, das die Deklarationen und Dokumentationskommentare in einer Gruppe von Quellendateien syntaktisch analysiert und eine Reihe von HTML-Seiten erstellt, die die Klassen, untergeordneten Klassen, Schnittstellen, Konstruktoren, Methoden und Felder beschreiben.

Java EE. Siehe Java Platform Enterprise Edition.

Java EE-Anwendung. Eine beliebige implementierbare Einheit mit Java EE-Funktionalität. Bei dieser Einheit kann es sich um ein einzelnes Modul oder um eine Gruppe von Modulen handeln, die in einer EAR-Datei (EAR = Enterprise Archive) mit einem Java EE-Anwendungsimplementierungsdeskriptor gepackt sind. (Sun)

Java EE Connector Architecture (JCA). Eine Standardarchitektur für die Verbindung der Java EE-Plattform mit heterogenen unternehmensweiten Informationssystemen (EIS = Enterprise Information Systems).

Java-EE-Server. Eine Laufzeitumgebung, die EJB- oder Web-Container bereitstellt.

Java-Klasse. Eine Klasse, die in der Programmiersprache Java geschrieben wurde.

JavaMail API. Ein plattform- und protokollunabhängiges Gerüst für die Erstellung Java-basierter Mail-Clientanwendungen.

Java Message Service (JMS). Eine Anwendungsprogrammierschnittstelle, die Java-Sprachfunktionen für die Verarbeitung von Nachrichten zur Verfügung stellt.

Java Naming and Directory Interface (JNDI). Eine Erweiterung der Java-Plattform, die eine Standardschnittstelle für heterogene Namens- und Verzeichnisservices bereitstellt.

Java Platform, Enterprise Edition (Java EE). Eine Umgebung zum Entwickeln und Implementieren von Unternehmensanwendungen, die von Sun Microsystems Inc. definiert wurde. Die Java EE-Plattform besteht aus einer Reihe von Services, Anwendungsprogrammierschnittstellen (APIs) und Protokollen, die die Funktionalität für die Entwicklung mehrschichtiger, webbasierter Anwendungen zur Verfügung stellen. (Sun)

Java Platform, Standard Edition (Java SE). Die zentrale Plattform der Java-Technologie. (Sun)

Java-Projekt. Bei Eclipse ein Projekt, das kompilierbaren Java-Quellcode enthält und einen Container für Quellenordner oder Pakete darstellt.

Java Runtime Environment (JRE). Eine Untergruppe des Java Developer Kit, die die zentralen ausführbaren Programme und Dateien enthält, auf denen die Java-Standardplattform basiert. Die Java Runtime Environment (JRE) umfasst die Java Virtual Machine (JVM) sowie die wichtigsten Klassen und Unterstützungsdateien.

JavaScript. Eine Web-Scripting-Sprache, die sowohl von Browsern als auch von Web-Servern verwendet wird. (Sun)

JavaScript Object Notation. Ein einfaches Datenaustauschformat, das auf der Objekt-Literal-Notation von JavaScript basiert. JSON ist programmiersprachenneutral, verwendet allerdings Konventionen aus Sprachen, wie C, C++, C#, Java, JavaScript, Perl und Python.

Java SE. Siehe Java Platform Standard Edition.

Java Secure Socket Extension (JSSE) . Ein Java-Paket, das zur Bereitstellung der sicheren Internetkommunikation dient. Es implementiert eine Java-Version der Protokolle SSL (Secure Sockets Layer) und TSL (Transport Layer Security) und unterstützt die Datenverschlüsselung, die Serverauthentifizierung sowie die Überprüfung der Nachrichtenintegrität und optional die Clientauthentifizierung.

Java SE Development Kit (JDK). Der Name des Software Development Kit, das von Sun Microsystems für die Java-Plattform bereitgestellt wird.

JavaServer Pages (JSP) . Eine serverseitige Scripting-Technologie, mit deren Hilfe Java-Code dynamisch in Webseiten (HTML-Dateien) eingebettet und beim Bereitstellen der Seite ausgeführt werden kann, um dynamische Inhalte an den Client zurückzugeben.

Java Specification Request (JSR). Eine formal eingereichter Spezifikationsvorschlag für die Java-Plattform.

Java Virtual Machine (JVM). Die Softwareimplementierung eines Prozessors, die kompilierten Java-Code (Applets und Anwendungen) ausführt.

Java Virtual Machine Profiler Interface (JVMPI). Ein Profilerstellungstool, das die Erfassung von Informationen wie z. B. von Daten zur Garbage-Collection sowie Angaben zu der Java Virtual Machine-Anwendungsprogrammierschnittstelle (JVM-API) unterstützt, die zur Ausführung des Anwendungsservers eingesetzt wird.

JAX . Siehe Java API for XML.

JCA. Siehe Java EE Connector Architecture.

JDBC. Siehe Java Database Connectivity.

JDK. Siehe Java SE Development Kit.

JMS. Siehe Java Message Service.

JMS-Datenbindung. Eine Datenbindung, die eine Zuordnung zwischen dem von einer externen JMS-Nachricht verwendeten Format und der SDO-Darstellung (SDO = Service Data Object) bereitstellt, die von einem SCA-Modul (SCA = Service Component Architecture) eingesetzt wird.

JMX . Siehe Java Management Extensions.

JMX (Java Management Extensions). Eine Methode für die Verwaltung mit Java-Technologie. JMX ist eine universelle, offene Erweiterung der Programmiersprache Java für die Verwaltung, die in jedem Unternehmen eingesetzt werden kann, in dem eine Verwaltung erforderlich ist.

JNDI. Siehe Java Naming and Directory Interface.

JSP. Siehe JavaServer Pages.

JSP-Datei. Eine scriptgesteuerte HTML-Datei mit der Dateierweiterung ".jsp", die den Einschluss dynamischer Inhalte in Webseiten ermöglicht. Eien JSP-Datei kann über einen URL direkt angefordert, von einem Servlet oder in einer HTML-Seite aufgerufen werden.

JSP-Seite. Ein textbasiertes Dokument, das feste Schablonendaten und JSP-Elemente verwendet, die beschreiben, wie eine Anforderung verarbeitet werden soll, um eine Antwort zu erstellen. (Sun)

JSR. Siehe Java Specification Request.

JSSE. Siehe Java Secure Socket Extension.

JVM. Siehe Java Virtual Machine.

JVMPI. Siehe Java Virtual Machine Profiler Interface.

Jython. Eine Implementierung der Programmiersprache Python, die mit der Java-Plattform integriert ist.

Katalog. Ein Container, dessen Projektbaum abhängig vom jeweiligen Containertyp Prozesse, Daten, Ressourcen, Organisationen oder Berichte enthält.

Katalogservice. Ein Service, der die Positionierung von Shards steuert und den Status der Container erkennt und überwacht.

Kategorie. Ein Container, der in einem Strukturdiagramm zum Gruppieren von Elementen auf der Basis eines gemeinsamen Attributs oder einer gemeinsamen Qualität verwendet wird.

Klasse. In der objektorientierten Programmierung ein Modell oder eine Schablone, die verwendet werden kann, um Objekte mit einer gemeinsamen Definition und gemeinsamen Merkmalen, Operationen und Verhaltensmustern zu erstellen. Ein Objekt ist eine Instanz einer Klasse.

Klassendatei. Eine kompilierte Java-Quellendatei.

Klassenhierarchie. Die Beziehungen zwischen Klassen, die einen gemeinsamen Vorfahren (Basisklasse) haben.

Klassenlader. Eine Komponente der Java Virtual Machine (JVM), die für das Suchen und Laden von Klassendateien verwendet wird. Ein Klassenladeprogramm wirkt sich auf das Packen von Anwendungen und das Laufzeitverhalten von gepackten Anwendungen aus, die auf Anwendungsservern implementiert sind.

Klassenpfad. Eine Liste mit Verzeichnissen und JAR-Dateien, die Ressourcendateien oder Java-Klassen enthalten, die ein Programm zur Laufzeit dynamisch laden kann.

Klassifikationsmerkmal. Ein spezielles Attribut, das zur Gruppierung und Farbcodierung von Prozesselementen dient.

Knoten.

1. Eine logische Gruppierung verwalteter Server.

2. Ein Element in einer Baumstruktursteuerung, wie beispielsweise ein einfaches Element, ein Verbundelement, ein Zuordnungsbefehl, ein Kommentar oder ein Gruppenknoten.

3. In XML die kleinste Einheit der gültigen und vollständigen Struktur in einem Dokument.

4. Die grundlegenden Formen, aus denen sich ein Diagramm zusammensetzt.

Kohärenter Cache. Ein Cache, der die Integrität so verwaltet, dass alle Clients dieselben Daten sehen.

Kompiliereinheit. Ein Teil eines Computerprogramms, das ausreichend vollständig ist, um ordnungsgemäß kompiliert werden zu können.

Kompilierzeit. Der Zeitraum, während dessen ein Computerprogramm zu einem ausführbaren Programm kompiliert wird.

Komponente.

1. Ein wiederverwendbares Objekt oder Programm, das eine bestimmte Funktion ausführt und mit anderen Komponenten und Anwendungen arbeitet.

2. Bei Eclipse ein bestimmtes Plug-in oder mehrere Plug-ins, die zusammenarbeiten, um eine eigenständige Funktionsgruppe bereitzustellen.

Komponentenelement. Eine Entität in einer Komponente, für die ein Unterbrechungspunkt gesetzt werden kann, wie beispielsweise eine Aktivität oder ein Java-Snippet in einem Geschäftsprozess bzw. ein Mediationsbasiselement oder ein Knoten in einem Mediationsablauf.

Komponenteninstanz. Eine aktive Komponente, die parallel zu anderen Instanzen derselben Komponente ausgeführt werden kann.

Komponententest. Ein automatisierter Test einer oder mehrerer Komponenten einer Unternehmensanwendung, die Java-Klassen, EJB-Beans oder Web-Services beinhalten kann.

Lastausgleich. Die Überwachung von Anwendungsservern und die Verwaltung der Workload auf Servern. Wenn einer der Server überlastet ist, werden die Anforderungen an einen anderen Server mit mehr Kapazität weitergeleitet.

Laufzeit. Der Zeitraum, während dessen ein Computerprogramm ausgeführt wird.

Laufzeittopologie. Eine Abbildung des momentanen Zustands der Umgebung.

LDAP. Siehe Lightweight Directory Access Protocol.

LDAP-Verzeichnis. Ein Repository-Typ, in dem Informationen zu Personen, Organisationen und anderen Ressourcen gespeichert werden und auf den über das LDAP-Protokoll zugegriffen wird. Die Einträge im Repository sind in einer hierarchischen Struktur organisiert. In manchen Fällen gibt diese hierarchische Struktur die Struktur oder Geographie einer Organisation wieder.

Lebensdauer. (TTL, Time-to-Live) Das Zeitintervall in Sekunden, für das ein Eintrag vor dem Löschen im Cache enthalten sein kann.

Lebenszyklus. Ein vollständiger Durchlauf durch die vier Softwareentwicklungsphasen Konzeption, Ausarbeitung, Konstruktion und Ablösung.

Lightweight Directory Access Protocol (LDAP). Ein offenes Protokoll, das TCP/IP verwendet, um den Zugriff auf Verzeichnisse bereitzustellen, die ein X.500-Modell unterstützen und bei dem weniger aufwendige Ressourcenanforderungen gelten als bei dem komplexeren X.500 Directory Access Protocol (DAP). LDAP kann beispielsweise eingesetzt werden, um Personen, Organisationen und andere Ressourcen in einem Internet- oder Intranetverzeichnis zu lokalisieren.

Listener. Ein Programm, das ankommende Anforderungen erkennt und den zugehörigen Channel startet.

Listener-Port. Ein Objekt, das die Zuordnung zwischen einer Verbindungs-Factory, einer Zieladresse und einer implementierten Message-Driven-Bean (MDB) definiert. Listener-Ports vereinfachen die Verwaltung der Zuordnungen zwischen diesen Ressourcen.

Loader. (Ladeprogramm) Eine Komponente, die Daten aus einem persistenten Speicher liest bzw. in diesen schreibt.

Lokal.

1. Bezeichnet eine Einheit, eine Datei oder ein System, auf die bzw. das direkt über das System eines Benutzers zugegriffen wird, ohne dass hierbei eine Übertragungsleitung verwendet werden muss.

2. Bezeichnet ein Element, das nur innerhalb seines eigenen Prozesses zur Verfügung steht.

Lokale Datenbank. Eine Datenbank, die sich auf der momentan verwendeten Workstation befindet.

LTPA. Siehe Lightweight Third Party Authentication.

LTPA (Lightweight Third Party Authentication). Ein Protokoll, das die Sicherheit in einer verteilten Umgebung durch Verschlüsselung unterstützt.

Managed Bean (MBean). In der Spezifikation Java Management Extensions (JMX) die Java-Objekte, die Ressourcen und die zugehörige Instrumentierung implementieren.

Map.

1. Eine Datenstruktur, die Schlüssel Werten zuordnet.

2. Eine Datei, die die Umsetzung zwischen Quellen und Zielen definiert.

3. In der EJB-Entwicklungsumgebung die Spezifikation, die angibt, wie die CMP-Felder (CMP = Container-managed Persistence) einer Enterprise-Bean den Spalten in einer Tabelle einer relationalen Datenbank oder einem anderen persistenten Speicher entsprechen.

MBean. Siehe Managed Bean.

MBean-Provider. Eine Bibliothek, die eine Implementierung einer JMX-MBean und die zugehörige XML-Deskriptordatei enthält.

Messgröße. Ein Behältnis für Informationen in einem Überwachungskontext, normalerweise für einen Geschäftsleistungsmesswert.

Methode. Bei der objektorientierten Programmierung eine Operation, die von einem Objekt ausgeführt werden kann. Ein Objekt kann viele Methoden aufweisen.

Nachgeordnet. Dieser Begriff bezeichnet die Richtung eines Verarbeitungsablaufs. Dieser verläuft vom ersten Knoten innerhalb des Prozesses (d. h. vom vorgeordneten Knoten) zum letzten Knoten des Prozesses (d. h. zum nachgeordneten Knoten).

Namespace. Ein logischer Container, in dem alle Namen eindeutig sind. Die eindeutige Kennung für ein Artefakt setzt sich aus dem Namespace und dem lokalen Namen des Artefakts zusammen.

Node Agent. Ein Verwaltungsagent, der alle Anwendungsserver auf einem Knoten verwaltet und den Knoten in der Verwaltungszelle repräsentiert.

ObjectGrid. Eine Grid-fähige Speicherdatenbank für Anwendungen, die in Java geschrieben werden. ObjectGrid kann als speicherinterne Datenbank oder zum Verteilen von Daten in einem Netz verwendet werden.

Object Request Broker. In der objektorientierten Programmierung eine Software, die als Vermittler fungiert, indem Sie den Austausch von Anforderungen und Antworten zwischen Objekten transparent erlaubt.

Objekt. Im objektorientierten Design oder in der objektorientierten Programmierung eine konkrete Ausführung (Instanz) einer Klasse, die aus Daten und den zugehörigen Operationen besteht. Ein Objekt enthält die Instanzdaten, die von der Klasse definiert werden. Eigner der Operationen, die den Daten zugeordnet sind, ist aber die Klasse.

Objektorientierte Programmierung. Eine Programmierungsmethode auf der Basis der Konzepte zur Datenabstraktion und Vererbung. Im Gegensatz zu Verfahren der prozeduralen Programmierung liegt der Schwerpunkt der objektorientierten Programmierung nicht darauf, wie etwas erreicht wird, sondern darauf, welche Datenobjekte das Problem umfasst und wie diese bearbeitet werden.

ODBC. Siehe Open Database Connectivity.

Open Database Connectivity (ODBC). Eine standardisierte Anwendungsprogrammierschnittstelle (API = Application Programming Interface) für den Zugriff auf Daten in relationalen und nicht relationalen Datenbankmanagementsystemen. Unter Verwendung dieser API können Datenbankanwendungen auf Daten zugreifen, die in Datenbankmanagementsystemen auf verschiedenen Computern gespeichert sind, auch wenn die einzelnen Datenbankmanagementsysteme unterschiedliche Formate für die Datenspeicherung und unterschiedliche Programmierschnittstellen verwenden.

Open Source. Software, deren Quellcode für die Verwendung oder für Änderungen allgemein zur Verfügung steht. Open-Source-Software wird normalerweise in Form einer allgemeinen Zusammenarbeit entwickelt und wird frei verfügbar gemacht, ihre Verwendung und Neuverteilung kann jedoch Lizenzbeschränkungen unterliegen. Linux ist ein sehr bekanntes Beispiel für Open-Source-Software.

Operation. Eine Implementierung von Funktionen oder Abfragen, zu deren Ausführung ein Objekt aufgerufen werden kann.

ORB. Siehe Object Request Broker.

Ordner. Ein Container, der zum Verwalten von Objekten verwendet wird.

Organisation. Eine Entität, in der Personen zusammenarbeiten, um angegebene Ziele zu erreichen, wie ein Unternehmen, eine Firma oder eine Fabrik.

Paket.

1. In der Java-Programmierung ein Gruppe von Typen. Pakete werden mit dem Schlüsselwort package deklariert. (Sun)

2. Der Wrapper um den Dokumentinhalt, der das zum Übertragen eines Dokuments im Internet zu verwendende Format (z. B. RNIF, AS1 oder AS2) definiert.

3. In Module zusammengefasste Komponenten und in Unternehmensanwendungen zusammengefasste Module.

Partitionierungsfeature (WPF). Ein Programmierungsframework und eine Systemmanagementinfrastruktur, die das Konzept der Partitionierung für Enterprise-Beans, HTTP-Datenverkehr und Datenbankzugriff unterstützt.

Performance Monitoring Infrastructure (PMI). Eine Gruppe von Paketen und Bibliotheken für die Erfassung, Zustellung, Verarbeitung und Anzeige von Leistungsdaten.

Persistenter Datenspeicher. Ein nicht flüchtiger Speicher für Ereignisdaten (wie beispielsweise ein Datenbanksystem), der über Sitzungsgrenzen hinweg existiert und nach der Ausführung des erstellenden Programms oder Prozesses erhalten bleibt.

Persistent speichern. Elemente über Sitzungsgrenzen hinweg verwalten, in der Regel in einem nicht flüchtigen Speicher wie einem Datenbanksystem oder einem Verzeichnis.

Persistenz.

1. Ein Merkmal von Daten, die über Sitzungsgrenzen hinweg beibehalten werden, oder eines Objekts, das auch nach der Ausführung des erstellenden Programms oder Prozesses erhalten bleibt (normalerweise in einem nicht flüchtigen Speicher, wie einem Datenbanksystem).

2. In Java EE das Protokoll, mit dem der Status einer Entity-Bean zwischen den jeweiligen Instanzvariablen und einer zugrunde liegenden Datenbank übertragen wird. (Sun)

Pessimistisches Sperren. Eine Sperrstrategie, bei der bei Auswahl einer Zeile eine Sperre gesetzt wird, die so lange gehalten wird, bis versucht wird, eine Aktualisierungs- oder Löschoperation mit Suche für diese Zeile durchzuführen.

Plattform Java. Ein Sammelbegriff für die Sprache Java zum Schreiben von Programmen. Sie umfasst eine Gruppe von APIs, Klassenbibliotheken und anderen Programmen, die bei der Entwicklung, Kompilierung und Prüfung von Programmen auf Fehler verwendet werden, sowie eine Java Virtual Machine (JVM), die die Klassendateien lädt und ausführt. (Sun)

Plug-in. Ein separat installierbares Softwaremodul, das einem vorhandenen Programm, einer vorhandenen Anwendung oder einer vorhandenen Schnittstelle eine Funktion hinzufügt.

PMI . Siehe Performance Monitoring Infrastructure.

Port. Gemäß Definition in einem WSDL-Dokument (Web Services Description Language) ein Endpoint, der als Kombination von Bindung und Netzadresse definiert wird.

Portnummer. Bei der Internetkommunikation die Kennung für einen logischen Connector zwischen einer Anwendungsentität und dem Transportservice.

Primärschlüssel.

1. Ein Objekt, das eine Entity-Bean eines bestimmten Typs eindeutig kennzeichnet.

2. In einer relationalen Datenbank ein Schlüssel, der eine einzige Zeile in einer Datenbanktabelle eindeutig kennzeichnet.

Primitiver Datentyp. In Java eine Datentypkategorie, die eine Variable mit einem Einzelwert beschreibt, dessen Größe und Format dem Typ entsprechen: eine Zahl, ein Zeichen oder ein boolescher Wert. Beispiele für primitive Datentypen sind byte, short, int, long, float, double, char, boolean.

Profil. Daten, die die Merkmale eines Benutzers, einer Gruppe, einer Ressource, eines Programms, einer Einheit oder einer fernen Position beschreiben.

Protokollbindung. Eine Bindung, mit der der Enterprise Service Bus Nachrichten unabhängig vom Übertragungsprotokoll verarbeiten kann.

Protokollierung. Die Aufzeichnung von Daten zu bestimmten Ereignissen auf dem System, wie z. B. Fehler.

Proxy. Ein Anwendungsgateway zwischen zwei Netzen für eine bestimmte Netzanwendung, wie Telnet oder FTP, wenn beispielsweise der Proxy-Telnet-Server einer Firewall die Authentifizierung des Benutzers ausführt und dann den Datenverkehr über den Proxy laufen lässt, als ob dieser nicht vorhanden wäre. Die Funktion wird in der Firewall ausgeführt, nicht auf der Client-Workstation. Dadurch vergrößert sich die Arbeitslast in der Firewall.

Proxy-Cluster. Eine Gruppe von Proxyservern, die HTTP Anforderungen über den Cluster verteilt.

Proxy-Peer-Zugriffspunkt. Ein Mittel, mit dem die Kommunikationseinstellungen für einen Peer-Zugriffspunkt ermittelt werden können, auf den kein direkter Zugriff möglich ist.

Proxy-Server.

1. Ein Server, der als Zwischenstation für HTTP-Webanforderungen von einer Anwendung oder einem Webserver auftritt. Ein Proxy-Server tritt stellvertretend für die Content-Server im Unternehmen auf.

2. Ein Server, der Anforderungen für einen anderen Server empfängt und für den Client agiert (als Proxy des Clients), um den angeforderten Service abzurufen. Ein Proxy-Server wird häufig verwendet, wenn der Client und der

Server für eine Direktverbindung inkompatibel sind. Beispiel: Der Client kann die Anforderungen zur Sicherheitsauthentifizierung des Servers nicht erfüllen, soll aber für einige Services berechtigt werden.

Prozess.

1. Eine immer weiter voranschreitende Prozedur, die aus einer Reihe gesteuerter Aktivitäten besteht, die systematisch auf ein bestimmtes Ergebnis oder Ziel gerichtet sind.

2. Die Reihenfolge der Dokumente oder Nachrichten, die zwischen den Community-Managern und den Teilnehmern ausgetauscht werden sollen, um eine Geschäftstransaktion auszuführen.

PTF. Siehe Vorläufige Programmkorrektur (Program Temporary Fix).

public.

1. Bei der objektorientierten Programmierung bezieht sich dieser Begriff auf einen Klasseneintrag, auf den alle Klassen zugreifen können.

2. In der Programmiersprache Java bezieht sich dieser Begriff auf eine Methode oder Variable, auf die Elemente aus anderen Klassen zugreifen können. (Sun)

Punkt-zu-Punkt. Dieser Begriff bezeichnet einen Stil einer Messaging-Anwendung, bei der die sendende Anwendung das Ziel der Nachricht kennt.

QoS. Siehe Servicequalität (Quality of Service).

Qualifikationsmerkmal. Ein einfaches Element, das einem anderen generischen Verbundelement oder einfachen Element eine bestimmte Bedeutung gibt. Qualifikationsmerkmale werden bei der Zuordnung einmaliger oder mehrfacher Vorkommen verwendet. Ein Qualifikationsmerkmal kann auch zur Bezeichnung des Namensbereichs verwendet werden, mit dem der zweite Teil des Namens interpretiert wird. Dieser Teil wird im Allgemeinen als ID bezeichnet.

Read-Through-Cache. Ein Teilcache, der Dateneinträge nach Schlüssel lädt, wenn sie angefordert werden. Wenn Daten nicht im Cache gefunden werden, werden die fehlenden Daten mit dem Ladeprogramm abgerufen, das die Daten aus dem Back-End-Repository lädt und in den Cache einfügt.

Region . Ein zusammenhängender Bereich des virtuellen Speichers, der allgemeine Merkmale aufweist und von Prozessen gemeinsam genutzt werden kann.

Rekursion. Eine Programmiertechnik, bei der ein Programm oder eine Routine sich selbst aufruft, um aufeinander folgende Schritte in einer Operation auszuführen, wobei jeder Schritt die Ausgabe des vorhergehenden Schrittes verwendet.

Replikat. Ein Server, der eine Kopie des Verzeichnisses bzw. der Verzeichnisse eines anderen Servers enthält. Replikate sichern Server, um die Leistung bzw. Antwortzeiten zu verbessern und die Datenintegrität zu gewährleisten.

Replikation. Der Prozess, mit dem ein definierter Satz von Daten an mehr als einer Position verwaltet wird. Zur Replikation gehören das Kopieren festgelegter Änderungen für eine Position (eine Quelle) an eine andere (ein Ziel) und die Synchronisation der Daten an beiden Positionen.

Ressource.

1. Eine diskrete Ressource. Beispiele: Anwendungssuiten, Anwendungen, Geschäftsservices, Schnittstellen, Endpunkte und Geschäftsereignisse.

2. Eine Einrichtung eines Computer- oder Betriebssystems, die für einen Job, eine Task oder ein aktives Programm erforderlich ist. Ressourcen können unter anderem Hauptspeicher, Ein-/Ausgabeeinheiten, die Verarbeitungseinheit, Datensätze, Dateien, Bibliotheken, Ordner, Anwendungsserver und Steuer- oder Verarbeitungsprogramme sein.

3. Eine Person, ein Bauteil oder Material, die bzw. das für die Ausführung einer Task oder eines Projekts verwendet wird. Jede Ressource ist ein bestimmtes Vorkommen oder Beispiel für eine Ressourcendefinition.

Richtlinie. Eine Gruppe von Aspekten, die das Verhalten einer verwalteten Ressource oder eines Benutzers beeinflussen.

Rolle.

1. Eine Beschreibung einer Funktion, die von einer Einzelperson oder einer Massenressource ausgeführt werden soll, sowie der zu ihrer Ausführung erforderlichen Qualifikationen. Bei der Simulation und Analyse wird der Begriff Rolle auch für die qualifizierten Ressourcen verwendet.

2. Eine Jobfunktion, die die Tasks, die ein Benutzer ausführen kann, und die Ressourcen angibt, auf die ein Benutzer Zugriff hat. Einem Benutzer können eine oder mehrere Rollen zugeordnet werden.

3. Eine logische Gruppe von Principals, die eine Gruppe von Berechtigungen bereitstellt. Der Zugriff auf Operationen wird über die Zugriffsberechtigungen für eine Rolle gesteuert.

4. In einer Beziehung legt eine Rolle die Funktion und Teilnahme von Entitäten fest. Rollen erfassen Anforderungen bezüglich Struktur und Integritätsbedingungen für teilnehmende Entitäten und ihre Art der Teilnahme. Beispielsweise lauten die Rollen in einer Beschäftigungsbeziehung ″Arbeitgeber″ und ″Mitarbeiter″.

Root. Der Benutzername für den Systembenutzer mit der höchsten Berechtigungsstufe.

Schleife. Eine Instruktionsfolge, die wiederholt ausgeführt wird.

Schlüssel.

1. Ein verschlüsselter mathematischer Wert, der zum digitalen Signieren, Überprüfen, Verschlüsseln oder Entschlüsseln einer Nachricht verwendet wird.

2. Die Informationen, die eine reale Entität, die von einem Überwachungskontext protokolliert wird, beschreiben und eindeutig identifizieren.

Schlüsselwort. Eines der vordefinierten Wörter einer Programmiersprache, einer Kunstsprache, Anwendung oder eines Befehls.

Schnittstelle. Eine Gruppe von Operationen, die verwendet werden, um den Service einer Klasse oder einer Komponente anzugeben.

Schwellenwert. Eine Einstellung, die für einen Interrupt in einer Simulation gilt, der definiert, wann eine Prozesssimulation auf der Basis einer Bedingung angehalten werden soll, die für einen angegebenen Anteil von Vorkommen eines bestimmten Ereignisses existiert.

Script. Eine Reihe von Befehlen, die in einer Datei zusammengefasst sind und durch die beim Ausführen der Datei eine bestimmte Funktion ausgeführt wird. Scripts werden während ihrer Ausführung interpretiert.

Scripting. Eine Art der Programmierung, bei der vorhandene Komponenten als Basis zum Erstellen von Anwendung wiederverwendet werden.

SDK. Siehe Software Development Kit.

Secure Socket Layer (SSL). Ein Sicherheitsprotokoll für den Schutz personenbezogener Daten bei der Datenübertragung. Mit SSL können Client/Server-Anwendungen auf eine Weise kommunizieren, die das Ausspionieren, die Manipulation von Daten während der Übertragung und die Nachrichtenfälschung verhindern soll.

Serialisierung. Bei der objektorientierten Programmierung das sequenzielle Schreiben von Daten aus dem Programmspeicher an ein Kommunikationsmedium.

Serialisierungsmethode. Eine Methode zur Konvertierung von Objektdaten in ein anderes Format, wie z. B. Binärformat oder XML.

Servant-Region . Ein zusammenhängender Bereich des virtuellen Speichers, der dynamisch gestartet wird, wenn die Arbeitslast größer wird, und automatisch gestoppt wird, wenn die Arbeitslast geringer wird.

Server. Ein Softwareprogramm oder ein Computer, das bzw. der Services für andere Softwareprogramme oder Computer bereitstellt. Siehe auch Host.

Servercluster. Eine Gruppe von Servern, die in der Regel auf unterschiedlichen physischen Maschinen installiert sind und mit denselben Anwendungen konfiguriert sind, aber als ein logischer Server zusammenarbeiten.

Service-Level-Agreement (SLA). Ein Vertrag zwischen einem Kunden und einem Serviceanbieter, der die Erwartungen hinsichtlich des Service-Levels, die Verfügbarkeit, Leistungswerte und andere messbare Zielsetzungen betreffend, angibt.

Servicequalität (Quality of Service). Eine Gruppe von Kommunikationsmerkmalen, die eine Anwendung erfordert. Die Servicequalität definiert bestimmte Werte für die Übertragungspriorität, die Stufe der Weiterleitungszuverlässigkeit und die Sicherheitsstufe.

Servlet. Ein Java-Programm, das in einem Webserver ausgeführt wird und die Funktionen des Servers durch Generierung dynamischen Inhalts als Reaktion auf Webclientanforderungen erweitert. Servlets werden häufig verwendet, um Datenbanken mit dem Web zu verbinden.

Setter-Methode. Eine Methode, mit der ein Wert einer Instanz- oder Klassenvariablen manipuliert werden kann. Mit dieser Methode kann ein anderes Objekt den Wert einer seiner Variablen ermitteln.

Shard. (Engl., Scherbe) Eine Instanz einer Partition. Ein Shard kann eine primäre Instanz oder ein Replikat sein.

Shell-Script. Ein Programm oder Script, das von der Shell eines Betriebssystems interpretiert wird.

Sicherheitsadministrator. Die Person, die den Zugriff auf Geschäftsdaten und Programmfunktionen steuert.

Sicherheitstoken. Die Darstellung von Anforderungen, die ein Client stellt. Beispiele für derartige Anforderungen sind Name, Kennwort, Identität, Schlüssel, Zertifikat, Gruppe, Berechtigung usw.

Sitzung.

1. Eine logische oder virtuelle Verbindung zwischen zwei Stationen, Softwareprogrammen oder Einheiten in einem Netz, die die Kommunikation und den Datenaustausch zwischen diesen beiden Elementen ermöglicht.

2. Eine Reihe von Anforderungen an ein Servlet, die von demselben Benutzer und demselben Browser stammen.

3. In Java EE ein Objekt, mit dem ein Servlet die Interaktion eines Benutzers mit einer Webanwendung über mehrere HTTP-Anforderungen hinweg protokollieren kann.

Sitzungsaffinität. Eine Methode für die Konfiguration von Anwendungen, in denen ein Client immer mit demselben Server verbunden ist. Diese Konfigurationen inaktivieren das Workload Management nach der ersten Verbindung, weil eine Clientanforderung immer an denselben Server weitergeleitet werden muss.

Skalierbarkeit. Die Erweiterungsmöglichkeit eines Systems, wenn Ressourcen, wie Prozessoren, Hauptspeicher oder Speicher, hinzugefügt werden.

SLA. Siehe Service-Level-Agreement.

Software Development Kit (SDK). Eine Sammlung von Tools, APIs und Dokumentation, die den Entwickler bei der Entwicklung von Software in einer bestimmten Maschinensprache oder für eine bestimmte Betriebsumgebung unterstützt.

Speicherverlust. Die Auswirkung eines Programms, das Referenzen auf Objekte verwaltet, die nicht mehr benötigt werden und deshalb freigegeben werden müssen.

Sperre. Eine Methode, mit der verhindert werden kann, dass von einem Anwendungsprozess vorgenommene nicht festgeschriebene Änderungen von einem anderen Anwendungsprozess wahrgenommen werden oder dass ein Anwendungsprozess Daten aktualisiert, auf die ein anderer Prozess gerade zugreift. Eine Sperre gewährleistet die Integrität der Daten, indem sie verhindert, dass gleichzeitig angemeldete Benutzer auf inkonsistente Daten zugreifen.

SQL. Siehe Structured Query Language.

SQL-Abfrage. Eine Komponente bestimmter SQL-Anweisungen, von der eine Ergebnistabelle angegeben wird.

SSL. Siehe Secure Sockets Layer.

SSL-Channel. Ein Typ von Channel in einer Transportkette, der der Transportkette ein SSL-Konfigurationsrepertoire zuordnet.

Stapelspeicher. Ein Bereich im Hauptspeicher, in dem gewöhnlich Informationen, wie temporäre Registerdaten, Werte von Parametern und Rückkehradressen von Unterroutinen, gespeichert werden und der auf dem Prinzip LIFO (Last in, First out) basiert.

Statisch. Ein Schlüsselwort der Programmiersprache Java, mit dem eine Variable als Klassenvariable definiert wird.

Structured Query Language (SQL) . Eine standardisierte Sprache für die Definition und Bearbeitung von Daten in einer relationalen Datenbank.

Synchroner Prozess. Ein Prozess, der durch den Aufruf einer Anforderungs-/Antwortoperation gestartet wird. Das Ergebnis des Prozesses wird von derselben Operation zurückgegeben.

Synchrones Replikat. Ein Shard, das Aktualisierungen im Rahmen der Transaktion für das primäre Shard empfängt, um die Datenkonsistenz zu gewährleisten, wodurch die Antwortzeit im Vergleich mit einem asynchronen Replikat verbessert werden kann.

Synchronisieren. Das Hinzufügen, Entfernen oder Ändern einer Funktion oder eines Artefakts, damit sie/es einer anderen Funktion bzw. einem anderen Artefakt entspricht.

Syntax. Die Regeln zum Erstellen eines Befehls oder einer Anweisung.

Systemanalytiker. Ein Spezialist, der für die Umsetzung von Geschäftsanforderungen in Systemdefinitionen und Lösungen verantwortlich ist.

TCP. Siehe Transmission Control Protocol.

TCP-Channel . Ein Typ von Channel in einer Transportkette, der Clientanwendungen ermöglicht, persistente Verbindungen in einem lokalen Netz (LAN) zu verwenden.

TCP/IP. Siehe Transmission Control Protocol/Internet Protocol.

TCP/IP-Überwachungsserver. Eine Laufzeitumgebung, die alle Anforderungen und Antworten zwischen einem Webbbrowser und einem Anwendungsserver sowie TCP/IP-Aktivitäten überwacht.

Thin Application Client. Eine einfache, für den Download verfügbare Java-Anwendungslaufzeit, die mit Enterprise-Beans interagieren kann.

Thin Client. Ein Client, der über wenig oder keine installierte Software, aber über Zugriff auf Software verfügt, die von zugeordneten Netzservern verwaltet und bereitgestellt wird. Ein Thin Client ist eine Alternative zu einem Client mit vollem Funktionsumfang, wie beispielsweise einer Workstation.

Thread. Ein Datenstrom von Maschineninstruktionen, der von einem Prozess gesteuert wird. Bei manchen Betriebssystemen ist ein Thread die kleinste Operationseinheit in einem Prozess. Mehrere Threads können gleichzeitig laufen und dabei unterschiedliche Jobs ausführen.

Thread-Konflikt. Eine Bedingung, bei der ein Thread auf eine Sperre oder ein Objekt wartet, die bzw. das von einem anderen Thread gehalten wird.

Tivoli Performance Viewer. Ein Java-Client, der die PMI-Daten (Performance Monitoring Infrastructure) von einem Anwendungsserver abruft und sie in verschiedenen Formaten anzeigt.

Token.

1. Eine Markierung, mit der der aktuelle Status einer Prozessinstanz während eines Simulationslaufs protokolliert wird.

2. Eine bestimmte Nachricht oder ein bestimmtes Bitmuster, die bzw. das die Berechtigung oder temporäre Kontrolle für die Übertragung von Daten in einem Netz angibt.

Topologie. Die physische oder logische Zuordnung der Position der Netzbetriebskomponenten oder Knoten in einem Netz. Bus, Ring, Stern und Baumstruktur sind gängige Beispiele für Netztopologien.

Transaktion. Ein Prozess, bei dem alle Datenänderungen, die während einer Transaktion vorgenommen werden, zusammen als Einheit festgeschrieben oder als Einheit rückgängig gemacht werden.

Transmission Control Protocol/Internet Protocol (TCP/IP). Eine Gruppe von standardisierten, nicht proprietären Übertragungsprotokollen, die über verschiedene Typen miteinander verbundener Netze zuverlässige durchgängige Verbindungen zwischen Anwendungen zur Verfügung stellt.

Transmission Control Protocol (TCP). Ein Übertragungsprotokoll, das im Internet sowie in allen Netzen verwendet wird, die die IETF-Standards (IETF = Internet Engineering Task Force) für netzübergreifende Protokolle verwenden. TCP bietet ein zuverlässiges Host-to-Host-Protokoll in DFV-Netzen mit Paketvermittlung und in miteinander verbundenen Systemen solcher Netze.

Trennen von Webservern. Eine Topologie, in der der Webserver physisch vom Anwendungsserver getrennt ist.

Truststore-Datei. Eine Schlüsseldatenbankdatei, die die öffentlichen Schlüssel für eine vertrauenswürdige Entität enthält.

Typ.

1. In der Java-Programmierung eine Klasse oder Schnittstelle.

2. In einem WSDL-Dokument ein Element, das Datentypdefinitionen enthält und hierfür ein Typsystem verwendet (z. B. XSD).

Überwachungssignal. Ein Signal, das eine Entität an eine andere sendet, um anzugeben, dass sie noch immer aktiv ist.

UDDI. Siehe Universal Description, Discovery, and Integration.

Umgebung. Eine benannte Gruppe von logischen und physischen Ressourcen, die verwendet wird, um die Leistung einer Funktion zu verbessern.

Umgebungsvariable. Eine Variable, die angibt, wie ein Betriebssystem oder ein anderes Programm ausgeführt wird, oder die die vom Betriebssystem erkannten Einheiten festlegt.

Unbeaufsichtigte Installation. Eine Installation, die keine Nachrichten an die Konsole sendet, sondern Nachrichten und Fehler in Protokolldateien speichert. Bei einer unbeaufsichtigten Installation können Antwortdateien für die Dateneingabe verwendet werden.

Unbeaufsichtigter Modus. Eine Methode für die Installation oder Deinstallation einer Produktkomponente über die Befehlszeile, ohne dass eine grafische Benutzerschnittstelle angezeigt wird. Bei Verwendung des Befehlszeilenmodus gibt der Benutzer die für das Installations- oder Deinstallationsprogramm erforderlichen Daten direkt in der Befehlszeile oder in einer Datei (mit dem Namen Options- oder Antwortdatei) an.

Uniform Resource Identifier (URI).

1. Eine kompakte Zeichenfolge zum Identifizieren einer abstrakten oder physischen Ressource.

2. Eine eindeutige Adresse, die zum Identifizieren von Inhalten im Web (z. B. einer Textseite, einem Video- oder Tonclip, einem Standbild oder animierten Bild bzw. einem Programm) verwendet wird. Die häufigste Form von URI ist die Adresse einer Webseite, die eine Sonderform oder eine Untergruppe der URI darstellt und Uniform Resource Locator (URL) genannt wird. Eine URI beschreibt in der Regel, wie auf die Ressource, den Computer mit der Ressource und den Namen der Ressource (einen Dateinamen) auf dem Computer zugegriffen wird.

Uniform Resource Locator (URL). Die eindeutige Adresse einer Informationsressource, die in einem Netz wie dem Internet zugänglich ist. Der URL enthält das Namenskürzel für das Protokoll, mit dem auf die Informationsressource zugegriffen wird, und die Informationen, mit deren Hilfe das Protokoll die Informationsressource lokalisiert.

Uniform Resource Name (URN). Ein Name, der einen Web-Service eindeutig für einen Client identifiziert.

Universal Description, Discovery, and Integration (UDDI). Eine Reihe standardisierter Spezifikationen, mit denen Unternehmen und Anwendungen auf schnelle und einfache Weise Web-Services über das Internet finden und verwenden können.

Universally Unique Identifier (UUID). Die numerische 128-Bit-Kennung, mit der sichergestellt wird, dass zwei Komponenten nicht dieselbe Kennung aufweisen.

UNIX System Services. Ein Element von z/OS, das eine UNIX-Umgebung erstellt, die den Spezifikationen XPG4 UNIX 1995 entspricht und zwei Schnittstellen offener Systeme für das Betriebssystem z/OS bereitstellt: eine Anwendungsprogrammierschnittstelle (API = Application Programming Interface) und eine interaktive Shell-Schnittstelle.

Unterabfrage. In SQL ein Subselect in einem Prädikat. Beispiel: Eine Anweisung SELECT in der WHERE- oder HAVING-Klausel einer anderen SQL-Anweisung.

Unterbrechungspunkt. Ein markierter Punkt innerhalb eines Prozesses oder eines Programmablaufs, bei dessen Erreichen der Ablauf angehalten wird, um beispielsweise Debug- oder Überwachungsoperationen auszuführen.

Untergeordneter Knoten. Ein Knoten, der sich im Geltungsbereich eines anderen Knotens befindet.

Unterklasse. In Java eine Klasse, die über Vererbung aus einer bestimmten Klasse ableitet wird.

Unternehmensanwendungsprojekt. Eine Struktur und Hierarchie von Ordnern und Dateien, die einen Implementierungsdeskriptor, ein IBM Erweiterungsdokument sowie Dateien enthalten, die für alle im Implementierungsdeskriptor definierten Java EE-Module einheitlich sind.

Unternehmensarchiv (EAR). Ein besonderer Typ einer JAR-Datei, der durch den Java EE-Standard definiert ist und der zur Implementierung von Java EE-Anwendungen auf Java EE-Anwendungsservern verwendet wird. Eine EAR-Datei enthält EJB-Komponenten, einen Implementierungsdeskriptor sowie Webarchivdateien (WAR-Dateien) für einzelne Webanwendungen. Siehe auch Webarchiv.

URI. Siehe Uniform Resource Identifier.

URL. Siehe Uniform Resource Locator.

URL-Schema. Ein Format, das eine andere Objektreferenz enthält.

URN. Siehe Uniform Resource Name.

UUID . Siehe Universally Unique Identifier.

Variable. Eine Darstellung eines änderbaren Werts.

Veraltet. Dieser Begriff bezeichnet eine Entität, z. B. ein Programmierungselement oder eine Programmierungsfunktion, die zwar noch unterstützt wird, deren Verwendung aber nicht mehr empfohlen wird und die möglicherweise sehr bald nicht mehr aktuell sein wird.

Vererbung. Eine Technik in der objektorientierten Programmierung, bei der vorhandene Klassen als Basis für die Erstellung anderer Klassen verwendet werden. Durch die Vererbung umfassen die spezifischeren Elemente die Struktur und das Verhalten der allgemeineren Elemente.

Verfügbarkeit.

1. Die Bedingung, die Benutzern den Zugriff auf und die Verwendung ihrer Anwendungen und Daten ermöglich.

2. Die Zeiträume, während derer auf eine Ressource zugegriffen werden kann. Die Services eines Vertragsnehmers können beispielsweise werktags von 09.00 Uhr bis 17.00 Uhr sowie samstags von 09.00 Uhr bis 15.00 Uhr verfügbar sein.

Verknüpfung.

1. (Join) Ein Prozesselement zur Rekombination und Synchronisation von Parallelverarbeitungspfaden nach einer Entscheidung oder Verzweigung. Eine Verknüpfung wartet an allen Eingangszweigen auf Eingabedaten, bevor die Fortführung des Prozesses zugelassen wird.

2. Eine relationale SQL-Operation, in der Daten aus zwei Tabellen, gewöhnlich über eine Verknüpfungsbedingung, die die zu verknüpfenden Spalten angibt, abgerufen werden können.

3. Die Konfiguration einer Eingangsverbindung, die das Verhalten dieser Verbindung festlegt.

Version. Ein gesondert lizenziertes Programm, das gewöhnlich neuen wichtigen Code oder neue wichtige Funktionen enthält.

Verteiltes eXtreme Scale. Ein Verwendungsmuster für die Interaktion mit eXtreme Scale, wenn Server und Clients in mehreren Prozessen existieren.

Verzweigung. (Fork) Ein Prozesselement, das Kopien seiner Eingabe erstellt und diese gleichzeitig über mehrere Verarbeitungspfade weiterleitet.

Virtualisierung. Ein Verfahren, das die Merkmale von Ressourcen auf der Basis der Art und Weise kapselt, in der andere Systeme mit diesen Ressourcen interagieren.

Virtuelle Maschine. Eine abstrakte Spezifikation einer Datenverarbeitungseinheit, die auf verschiedene Arten in der Software und Hardware implementiert sein kann.

Virtueller Host. Eine Konfiguration, mit der ein einziges Hostsystem mehreren Hostsystemen ähneln kann. Ressourcen, die einem virtuellen Host zugeordnet wurden, können nicht Daten mit Ressourcen gemeinsam nutzen, die einem anderen virtuellen Host zugeordnet wurden, auch wenn sich die virtuellen Hosts auf demselben physischen System befinden.

Vitalität. Der allgemeine Zustand bzw. Status der Datenbankumgebung.

Vorgeordnet. Dieser Begriff bezeichnet die Richtung des Verarbeitungsablaufs. Dieser verläuft vom Start des Prozesses (vorgeordnet) zum Ende des Prozesses (nachgeordnet).

Vorläufige Programmkorrektur (Program Temporary Fix). Bei den Produktreihen IBM System i, IBM System p und IBM System z eine Programmkorrektur, die von IBM getestet wurde und allen Kunden zur Verfügung gestellt wird. Siehe auch Fixpack.

Vorläufiger Fix. Ein zertifizierter Fix, der für alle Kunden zwischen regulär geplanten Fixpacks, Refresh-Packs oder Releases zur Verfügung gestellt wird. Siehe auch Fixpack.

Waiter. Ein Thread, der auf eine Verbindung wartet.

WAR . Siehe Webarchiv.

Wartungsmodus. Ein Status eines Knotens oder Servers, den Administratoren für dessen Diagnose, Wartung oder Optimierung verwenden können, ohne dass eingehender Datenverkehr in einer Produktionsumgebung unterbrochen wird.

WCCM . Siehe WebSphere Common Configuration Model.

Webarchiv (WAR). Ein komprimiertes Dateiformat, das vom Java EE-Standard definiert wird und mit dem alle Ressourcen, die zur Installation und Ausführung einer Webanwendung erforderlich sind, in einer einzigen Datei gespeichert werden können. Siehe auch Unternehmensarchiv.

Webbrowser. Ein Clientprogramm, das Anforderungen an einen Webserver einleitet und die Informationen anzeigt, die der Server zurückgibt.

Webcontainer. Ein Container, der den Webkomponentenvertrag der Java EE-Architektur implementiert. (Sun)

Webcontainer-Channel. Ein Typ von Channel in einer Transportkette, der eine Brücke in der Transportkette zwischen einem eingehenden HTTP-Channel und einer Servlet- oder JSP-Steuerkomponente erstellt.

Web-Crawler. Ein Crawler-Typ, der das Web durch Abrufen eines Webdokuments und Verfolgen der Links in diesem Dokument erkundet.

Webkomponente. Ein Servlet, eine JSP-Datei oder eine HTML-Datei. Eine oder mehrere Webkomponenten bilden ein Webmodul.

Webserver. Ein Softwareprogramm, das HTTP-Anforderungen verarbeiten kann.

Webserver-Plug-in. Ein Softwaremodul, das den Webserver bei der Weiterleitung von Anforderungen für dynamischen Inhalt, wie z. B. Servlets, an den Anwendungsserver unterstützt.

Website. Eine zusammengehörige Sammlung von im Web verfügbaren Dateien, die von einer einzigen Entität (einer Organisation oder einer Einzelperson) verwaltet wird und Hypertextinformationen für die Benutzer enthält. Eine Website enthält häufig Hypertext-Links zu anderen Websites.

WebSphere. Eine IBM Marke, die Tools für die Entwicklung von e-business-Anwendungen und Middleware für die Ausführung von Webanwendungen enthält.

WebSphere Common Configuration Model (WCCM) . Ein Modell, das programmgestützten Zugriff auf Konfigurationsdaten bietet.

What You See Is What You Get (WYSIWYG). Eine Funktionalität eines Editors, mit der Seiten ständig genau so angezeigt werden, wie sie gedruckt bzw. auf andere Weise wiedergegeben werden.

While-Schleife. Eine Schleife, die die gleiche Aktivitätenfolge so lange wiederholt, bis eine bestimmte Bedingung erfüllt ist. Die While-Schleife testet die zugehörige Bedingung zu Beginn jeder Schleife. Wenn die Bedingung von Beginn an nicht zutrifft, wird die Aktivitätenfolge in der Schleife nie ausgeführt.

WLM. Siehe Workload Manager.

Workload-Management. Die Optimierung der Verteilung eingehender Verarbeitungsaufträge an die Anwendungsserver, Enterprise-Beans, Servlets und anderen Objekte, die den jeweiligen Auftrag effektiv verarbeiten können.

Workload Manager (WLM). Eine Komponente von z/OS, mit der mehrere Workloads gleichzeitig in einem einzigen z/OS-Image oder über mehrere Images hinweg ausgeführt werden können.

Write-Behind-Cache. Ein Cache, der jede Schreiboperation asynchron über ein Ladeprogramm in die Datenbank schreibt.

Write-Through-Cache. (Durchschreibcache) Ein Cache, der jede Schreiboperation synchron über ein Ladeprogramm in die Datenbank schreibt.

WYSIWYG. Siehe What You See Is What You Get.

XA. Eine bidirektionale Schnittstelle zwischen einem oder mehreren Ressourcenmanagern, die den Zugriff auf gemeinsam genutzte Ressourcen ermöglichen, und einem Transaktionsmanager, der Transaktionen überwacht und auflöst.

XML. Siehe Extensible Markup Language.

X/Open XA. Die Schnittstelle X/Open Distributed Transaction Processing XA. Ein empfohlener Standard für die Kommunikation bei verteilten Transaktionen. Der Standard definiert eine bidirektionale Schnittstelle zwischen Ressourcenmanager, die den Zugriff auf gemeinsam genutzte Ressourcen in Transaktionen ermöglichen, und einem Transaktionsservice, der Transaktionen überwacht und auflöst.

Zeichenfolge. In Programmiersprachen das Datenformat, das für das Speichern und Bearbeiten von Text verwendet wird.

Zeitbindung. Eine angepasste Validierungsaktion, die zum Messen der Dauer eines Methodenaufrufs oder einer Folge von Methodenaufrufen verwendet wird.

Zeitgeber. Eine Task, die zu bestimmten Zeitpunkten eine Ausgabe generiert.

Zeitlimit. Ein Zeitintervall, das zugeordnet wird, damit ein Ereignis stattfinden oder beendet werden kann, bevor die Operation unterbrochen wird.

Zelle.

1. Eine Gruppe verwalteter Prozesse, die in denselben Deployment Manager eingebunden sind und Stammgruppen mit hoher Verfügbarkeit umfassen können.

2. Mindestens ein Prozess, der als Host für Laufzeitkomponenten eingesetzt wird. Jede Zelle hat mindestens eine benannte Stammgruppe.

Zertifikatssammelspeicher. Eine Gruppe vorläufiger Zertifikate oder Zertifikatwiderruflisten (CRL, Certificate Revocation List), die von einem Zertifikatpfad verwendet wird, um eine Zertifizierungskette für die Validierung zu erstellen.

Zonenbasierte Unterstützung. Eine Funktion, die die regelbasierte Positionierung von Shards ermöglicht, um die Grid-Verfügbarkeit durch Positionierung der Shards in unterschiedlichen Rechenzentren (auf verschiedenen Stockwerken oder sogar in verschiedenen Gebäuden oder Regionen) zu verbessern.

z/OS. Ein IBM Betriebssystem für Großrechner, bei dem 64-Bit-Realspeicher verwendet wird.

Bemerkungen

Hinweise auf IBM Produkte, Programme und Services in dieser Veröffentlichung bedeuten nicht, dass IBM diese in allen Ländern, in denen IBM vertreten ist, anbietet. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. An Stelle der IBM Produkte, Programme oder Services können auch andere ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb der Produkte, Programme oder Services in Verbindung mit Fremdprodukten und Fremdservices liegt beim Kunden, soweit solche Verbindungen nicht ausdrücklich von IBM bestätigt sind.

Für in diesem Dokument beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes
2, avenue Gambetta
92066 Paris La Defense
France

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation
Mail Station P300
522 South Road
Poughkeepsie, NY 12601-5400
USA
Attention: Information Requests

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Marken

Folgende Namen sind Marken der IBM Corporation in den USA und/oder anderen Ländern:

- AIX
- CICS
- Cloudscape
- DB2
- Domino
- IBM
- Lotus
- RACF
- Redbooks
- Tivoli
- WebSphere
- z/OS

Java und alle auf Java basierenden Marken und Logos sind Marken von Sun Microsystems, Inc. in den USA und/oder anderen Ländern.

LINUX ist eine Marke von Linus Torvalds in den USA und/oder anderen Ländern.

Microsoft, Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

UNIX ist eine eingetragene Marke von The Open Group in den USA und anderen Ländern.

Weitere Unternehmens-, Produkt- oder Servicenamen können Marken anderer Hersteller sein.

Index

K

L

M

N

O

P

R

S

T

U

V

W

X

Z

Gedruckt in Deutschland