WebSphere® eXtreme Scale Versão 7.0
Guia de Administração

Guia de Administração do WebSphere eXtreme Scale

WebSphere® eXtreme Scale Versão 7.0
Guia de Administração

Guia de Administração do WebSphere eXtreme Scale

Essa edição aplica-se à versão 7, release 0, do WebSphere eXtreme Scale e a todos os releases e modificações subsequentes, até que seja indicado de outra forma em novas edições.

Índice

Figuras v

Tabelas vii

Sobre o *Guia de Administração* ix

Capítulo 1. Introdução ao WebSphere eXtreme Scale 1

Capítulo 2. Planejando Implementação do Aplicativo 7
Configurações de Implementação para o eXtreme Scale 7
Requisitos de Hardware e Software 7
Ajuste do desempenho 8
Planejamento para Portas de Rede 8
Ajuste de Sistemas Operacionais e Rede 9
Propriedades do ORB e Configurações do Descritor de Arquivo 10
Ajuste da JVM para o WebSphere eXtreme Scale 11
Configurando a Detecção de Failover 13
Serviço de Catálogo de Alta Disponibilidade . . . 15
Lista de Verificação Operacional 17

Capítulo 3. Planejamento de Capacidade 21
Grades, Partições e Shards 21
Dimensionamento de Memória e Cálculo de Contagem de Partições 22
Dimensionando a CPU por Partição para Transações 24
Dimensionando CPUs para Transações Paralelas . . 25

Capítulo 4. Instalando e Implementando o WebSphere eXtreme Scale 27
Migrando para o WebSphere eXtreme Scale Versão 7.0 28
Instalando o WebSphere eXtreme Scale Independente 28
Usando o Object Request Broker com os Processos do WebSphere eXtreme Scale 30
Integrando o WebSphere eXtreme Scale com o WebSphere Application Server 30
Uso do Plug-in Installation Factory para Criação e Instalação de Pacotes Customizados 32
Criando e Alterando Perfis para o WebSphere eXtreme Scale 40
Instalando o WebSphere eXtreme Scale Silenciosamente 50
Parâmetros de Instalação 51
Usando o Update Installer para instalar os pacotes de manutenção 52
Configurando um Object Request Broker Customizado 53
Desinstalando o WebSphere eXtreme Scale 54

Capítulo 5. Customizando o WebSphere eXtreme Scale for z/OS . . 55
Instalando o WebSphere Customization Tools . . . 55
Gerando Definições de Customização 56
Fazendo Upload e Executando Tarefas Customizadas 57

Capítulo 6. Configurando WebSphere eXtreme Scale 59
Configurando um ObjectGrid Local de Memória . . 59
interface ObjectGrid 60
Interface BackingMap 63
Bloqueio de Entrada de Mapa 67
Configurando uma Grade Distribuída do eXtreme Scale. 70
Configurando um Cliente do eXtreme Scale . . . 72
Ativando o Mecanismo de Invalidação do Cliente 77
Configuração de Tempo Limite de Nova Tentativa de Solicitação 79
Resolução de Problemas de Configuração XML . . 81
Utilitários de Carga 86
Considerações sobre o Loader 88
Configurando Utilitários de Carga do JPA . . . 93
Configurando um Atualizador de Dados Baseado em Tempo do JPA 95
Plug-in do Cache JPA 96
Configuração do Plug-in do Cache JPA 100
Suporte de Armazenamento em Cache Write-behind 114
Configurando o Suporte Write-behind 119
Manipulando Atualizações Write-Behind Falhas 120
Configurando Evictores 125
Ativando o Evictor TTL 125
Conexão de um Evictor programável 130
Configurando o HashIndex 131
Configurando Replicação Ponto a Ponto com o JMS 133
Ativando o Mecanismo de Invalidação do Cliente 134
Distribuição de Alterações Entre Java Virtual Machines Peer 136
Listener de Eventos da JMS 139
Configurando com os Arquivos XML 142
Configuração de Topologia de Implementação 142
Arquivo XML descritor do ObjectGrid 150
Arquivo Descritor XML de Metadados de Entidade 171
Arquivo XML Descritor de Segurança 184
Referência de Arquivo de Propriedades 188
Arquivo de Propriedades do Servidor 189
Arquivo de Propriedades do Cliente. 195
Arquivo de Propriedades ORB 197
Integração com a Estrutura Spring 200

Beans de Extensão Spring e Suporte a Espaço de Nomes 201
Arquivo XML descritor do Spring 206
Usando o WebSphere Real Time 212

Capítulo 7. Administrando o Ambiente 215
Configurando a Disponibilidade de um ObjectGrid 215
Iniciando Servidores WebSphere eXtreme Scale Independentes 218
Iniciando o Serviço de Catálogo em um Ambiente Independente 218
Iniciando Processos do Contêiner 221
Script startOgServer 223
Configurando Portas TCP no Modo Independente 226
Parando Servidores eXtreme Scale Independentes 228
Script stopOgServer 229
Administrando o WebSphere eXtreme Scale com o WebSphere Application Server 230
Iniciando o Processo do Serviço de Catálogo em um Ambiente WebSphere Application Server . . 230
Iniciando Processos do Contêiner em um Ambiente WebSphere Application Server . . . 231
Configurando portas Protocolo de Controle de Transmissões (TCP) em um ambiente WebSphere Application Server 234
Gerenciando de Sessões HTTP 234
Configurando o Gerenciador de Sessões do WebSphere eXtreme Scale para Trabalhar com o WebSphere Application Server 237
Parâmetros de inicialização do contexto do servlet 240
Usando o WebSphere eXtreme Scale para Gerenciamento de Sessão SIP 241
Configurando o gerenciador de sessões HTTP para trabalhar com o WebSphere Application Server Community Edition 242
Usando o WebSphere eXtreme Scale para a Replicação do Estado de Sessão no WebSphere Application Server Community Edition. . . . 244

Capítulo 8. Monitorando seu Ambiente de Implementação 247
Visão Geral de Estatísticas 247
Monitorando com a API de Estatísticas 249
Módulos Estatísticos 251
Monitorando o Desempenho com o PMI do WebSphere Application Server 253
Ativando a PMI 253
Recuperando Estatísticas PMI 255
Módulos PMI 257
Utilizando o Utilitário wsadmin 264
Utilizando beans gerenciados (MBeans) para administrar seu ambiente 265
Visão Geral de Uso do MBean 265
Utilizando o Utilitário de Amostra xsAdmin . . . 266
Ferramentas de Fornecedor 268
Monitorando com o IBM Tivoli Enterprise Monitoring Agent para WebSphere eXtreme Scale 268
Monitorando Aplicativos eXtreme Scale com o CA Wily Introscope 274
Monitorando o eXtreme Scale com o Hyperic HQ 277
Logs e Rastreio 279
Opções de Rastreio 281

Capítulo 9. Protegendo o Ambiente de Implementação. 285
Segurança de Grade 286
Ativando a Segurança Local 287
Autenticação de Cliente do Aplicativo 287
Autorização do Aplicativo Cliente 290
Transport Layer Security e Secure Sockets Layer 293
Autenticação de Grade 295
Segurança do Java Management Extensions (JMX) 296
Arquivo XML Descritor de Segurança 298
Integração de Segurança com o WebSphere Application Server 301
Início e Encerramento de Servidores Seguros do eXtreme Scale 302

Capítulo 10. Resolução de Problemas 305
Logs e Rastreio 305
Opções de Rastreio 307
Mensagens 309
Notas sobre o Release 309
Desempenho e Boas Práticas 310

Capítulo 11. Glossário. 313

Avisos 337

Marcas Registradas 339

Índice Remissivo 341

Figuras

1. Utilitário de Carga 86
2. Topologia Integrado do JPA 97
3. Topologia Particionada Integrada do JPA 98
4. Topologia Remota do JPA 99
5. Estados de Disponibilidade de um ObjectGrid 216
6. Topologia do Gerenciamento de Sessões HTTP com uma Configuração de Contêiner Remoto 236
7. Visão Geral de Estatísticas 247
8. Visão Geral do MBean 249
9. Estrutura do Módulo ObjectGridModule 257
10. Exemplo da Estrutura do Módulo ObjectGridModule 258
11. Estrutura do mapModule 259
12. Exemplo de Estrutura do Módulo mapModule 259
13. Estrutura do Módulo hashIndexModule 261
14. Exemplo da Estrutura do Módulo hashIndexModule 261
15. Estrutura agentManagerModule 262
16. Exemplo da Estrutura agentManagerModule 263
17. Estrutura queryModule 264
18. Exemplo da Estrutura queryModule QueryStats.jpg 264

Tabelas

1. Intervalos de Pulsações 13
2. Lista de Verificação Operacional 18
3. Arquivos de tempo de execução no diretório de instalação `/ObjectGrid/lib` 29
4. Arquivos de tempo de execução no diretório de instalação `/lib` 31
5. Métodos de Interface ObjectGrid 60
6. Algumas Opções de write-behind 116
7. Suporte para Índice de Intervalo 133
8. Propriedades Customizadas para Gerenciamento de Sessão SIP com ObjectGrid . 241
9. Autenticação de credencial nas configurações do cliente e do servidor 288
10. Protocolo de Transporte a Ser Utilizado nas Configurações de Transporte do Cliente e Transporte do Servidor 293

Sobre o *Guia de Administração*

O conjunto da documentação do WebSphere eXtreme Scale inclui três volumes que fornecem as informações necessárias para utilizar, programar e administrar o produto WebSphere eXtreme Scale.

Biblioteca do WebSphere eXtreme Scale

A biblioteca do WebSphere eXtreme Scale contém os seguintes livros:

- O *Guia de Administração* contém as informações necessárias para os administradores de sistema, incluindo como planejar implementações do aplicativo, planejar capacidade, instalar e configurar o produto, iniciar e parar servidores, monitorar o ambiente e proteger o ambiente.
- O *Guia de Programação* contém informações para desenvolvedores de aplicativos sobre como desenvolver aplicativos para o WebSphere eXtreme Scale utilizando as informações da API incluídas.
- O *Visão Geral do Produto* contém um visualização de alto nível dos conceitos do WebSphere eXtreme Scale, incluindo cenários de caso de uso e tutoriais.

Para fazer download dos manuais, vá para a Página da Biblioteca do WebSphere eXtreme Scale.

Também é possível acessar as mesmas informações nesta biblioteca no Centro de Informações do WebSphere eXtreme Scale.

Quem Deve Utilizar este Manual

Este manual é destinado principalmente para administradores de sistema, administradores de segurança e operadores de sistema.

Como este Manual Está Estruturado

O manual contém informações sobre os seguintes tópicos principais:

- **Capítulo 1** inclui informações sobre introdução.
- **Capítulo 2** inclui informações sobre o planejamento da implementação do aplicativo.
- **Capítulo 3** inclui informações sobre o planejamento de capacidade.
- **Capítulo 4** inclui informações sobre a instalação do produto.
- **Capítulo 5** inclui informações sobre a customização da plataforma z/OS.
- **Capítulo 6** inclui informações sobre a configuração do produto.
- **Capítulo 7** inclui informações sobre a administração do ambiente.
- **Capítulo 8** inclui informações sobre o monitoramento do ambiente de implementação.
- **Capítulo 9** inclui informações sobre a segurança do ambiente de implementação.
- **Capítulo 10** inclui informações sobre a resolução de problemas do ambiente.
- **Capítulo 11** inclui informações sobre o glossário do produto.

Obtendo Atualizações para este Manual

É possível obter as atualizações para esse manual ao fazer download da versão mais recente da Página da Biblioteca do WebSphere eXtreme Scale.

Como Enviar Seus Comentários

Entre em contato com a equipe de documentação. Você localizou o que precisava? O conteúdo era exato e completo? Envie seus comentários sobre esta documentação por e-mail para wasdoc@us.ibm.com.

Capítulo 1. Introdução ao WebSphere eXtreme Scale

Depois de instalar o WebSphere eXtreme Scale em um ambiente independente, use as seguintes etapas como uma simples introdução para sua capacidade como uma grade de dados em memória.

A instalação independente do WebSphere eXtreme Scale inclui uma amostra que pode ser usada para verificar a sua instalação e ver como uma grade e um cliente simples do eXtreme Scale podem ser usados. A amostra de iniciação está localizada no diretório *installRoot*/ObjectGrid/gettingstarted.

A amostra de iniciação é fornecida para uma introdução rápida à operação básica e de funcionalidade do eXtreme Scale. A amostra consiste de scritps de shell e lote projetados para iniciar uma grade simples com muito pouca customização necessária. Além disso, um programa cliente, incluindo a fonte, é fornecido para executar funções simples de criação, leitura, atualização e exclusão (CRUD) nesta grade básica.

Scripts e suas Funções

Essa amostra fornece os seguintes scripts:

O script env.sh|bat é chamado pelos outros scripts para configurar as variáveis necessárias do ambiente. Normalmente não é necessário alterar esse script.

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

O script runcat.sh|bat inicia o processo de serviço de catálogo do eXtreme Scale no sistema local.

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

O script runcontainer.sh|bat inicia um processo de servidor de contêiner. É possível executar esse script várias vezes com nomes do servidor exclusivos especificados para iniciar qualquer número de contêineres. Essas instâncias podem funcionar juntas para hospedar informações particionadas e redundantes na grade.

- UNIX Linux `./runcontainer.sh unique_server_name`
- Windows `runcontainer.bat unique_server_name`

O script runclient.sh|bat executa o cliente CRUD simples e inicia a operação especificada.

- UNIX Linux `./runclient.sh command value1 value2`
- Windows `runclient.sh command value1 value2`

Para *command*, use uma das seguintes opções:

- Especifique `i` para inserir *value2* na grade com a chave *value1*
- Especifique `u` para atualizar o objeto com chave pelo *value1* para o *value2*
- Especifique `d` para excluir o objeto com chave pelo *value1*

- Especifique `g` para recuperar e exibir o objeto com chave pelo *value1*

Nota: O arquivo *installRoot*`/ObjectGrid/ gettingstarted/src/Client.java` é o programa cliente que demonstra como se conectar a um servidor de catálogo, obter uma instância de ObjectGrid, e usar a API de ObjectMap.

Etapas Básicas

Use as seguintes etapas para iniciar sua primeira grade e executar um cliente para interagir com a grade.

1. Abra uma janela de sessão do terminal ou linha de comandos.
2. Utilize o seguinte comando para navegar para o diretório `gettingstarted`:

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

   Substitua *installRoot* pelo caminho para o diretório-raiz da instalação do eXtreme Scale ou o caminho do arquivo-raiz do trial do eXtreme Scale extraído *installRoot*.
3. Configure ou exporte a variável de ambiente JAVA_HOME para referenciar um diretório de instalação JDK ou JRE Versão 1.5 ou mais recente válido.

   UNIX Linux export JAVA_HOME=*Java_home_directory*

   Windows set JAVA_HOME=*Java_home_directory*
4. Execute o script a seguir para iniciar um processo de serviço de catálogo no host local:
   - UNIX Linux `./runcat.sh`
   - Windows `runcat.bat`

   O processo do serviço de catálogo executa na janela do terminal atual.
5. Abra outra janela de sessão de terminal ou de linha de comandos e execute o seguinte comando para iniciar uma instância do servidor de contêiner:
   - UNIX Linux `./runcontainer.sh server0`
   - Windows `runcontainer.bat server0`

   O servidor de contêiner será executado na janela do terminal atual. Repita as etapas 5 e 6 se desejar iniciar mais instâncias do servidor de contêiner para suportar a replicação.
6. Abra outra janela de sessão de terminal ou linha de comandos para executar comandos do cliente.
   - Incluir dados na grade:
     - UNIX Linux `./runclient.sh i key1 helloWorld`
     - Windows `runclient.bat i key1 helloWorld`
   - Procurar e exibir o valor:
     - UNIX Linux `./runclient.sh g key1`
     - Windows `runclient.bat g key1`
   - Atualizar o valor:
     - UNIX Linux `./runclient.sh u key1 goodbyeWorld`
     - Windows `runclient.bat u key1 goodbyeWorld`
   - Excluir o valor:
     - UNIX Linux `./runclient.sh d key1`

– Windows `runclient.bat d key1`

7. Use <ctrl+c> para parar o processo de serviço de catálogo e os servidores de contêiner em suas respectivas janelas.

Definindo um ObjectGrid

A amostra usa os arquivos `objectgrid.xml` e `deployment.xml` que estão no diretório *installRoot*`/ObjectGrid/gettingstarted/xml` para iniciar um servidor de contêineres. O arquivo `objectgrid.xml` é o arquivo XML descritor do ObjectGrid e o arquivo `deployment.xml` é o arquivo XML descritor de política de implementação do ObjectGrid. Ambos os arquivos juntos definem uma topologia de ObjectGrid distribuída.

Arquivo XML descritor do ObjectGrid

Um arquivo XML descritor de ObjectGrid é usado para definir a estrutura do ObjectGrid que será usada pelo aplicativo. Ele inclui uma lista de configurações de BackingMap. Esses BackingMaps são o armazenamento de dados atual para dados em cache. O exemplo a seguir é um arquivo `objectgrid.xml` de amostra. As primeiras linhas do arquivo incluem o cabeçalho obrigatório de cada arquivo XML do ObjectGrid. Este arquivo de exemplo define o Grid ObjectGrid com BackingMaps Map1 e 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>
```

Arquivo Descritor XML de Política de Implementação

Um arquivo XML descritor de política de implementação é passado para um servidor de contêineres ObjectGrid durante a inicialização. Uma política de implementação deve ser usada com o arquivo XML de ObjectGrid e deve ser compatível com o XML de ObjectGrid que é usado com ele. Para cada elemento objectgridDeployment na política de implementação, você deve ter um elemento ObjectGrid correspondente no XML do seu ObjectGrid. Os elementos de backingMap que são definidos dentro do elemento objectgridDeployment devem ser consistentes com os backingMaps localizados no XML de ObjectGrid. Cada backingMap deve ser referido dentro de um e apenas um mapSet.

O arquivo XML descritor de política de implementação deve igualar-se com o arquivo XML ObjectGrid correspondente, o arquivo `objectgrid.xml`. No seguinte exemplo, as primeiras linhas do arquivo `deployment.xml` incluem o cabeçalho obrigatório de cada arquivo XML de política de implementação. O arquivo define o elemento objectgridDeployment para o ObjectGrid da Grade que está definido no arquivo `objectgrid.xml`. Ambos os BackingMaps, Map1 e Map2, que são definidos dentro do ObjectGrid da Grade estão incluídos no mapSet que tem os atributos numberOfPartitions, minSyncReplicas e maxSyncReplicas configurados.

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

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

```
            <map ref="Map2"/>
        </mapSet>
    </objectgridDeployment>

</deploymentPolicy>
```

O atributo numberOfPartitions do elemento mapSet especifica o número de partições para o mapSet. Ele é um atributo opcional e o padrão é 1. O número deve ser apropriado para a capacidade antecipada da grade.

O atributo minSyncReplicas de mapSet especifica o número mínimo de réplicas síncronas para cada partição no mapSet. Ele é um atributo opcional e o padrão é 0. Primária e réplica não são colocadas até que o domínio possa suportar o número mínimo de réplicas síncronas. Para suporte do valor minSyncReplicas, é preciso de mais um contêiner do que o valor de minSyncReplicas. Se o número de réplicas síncronas ficar abaixo do valor de minSyncReplicas, grave transações que não são mais permitidas àquela partição.

O atributo maxSyncReplicas de mapSet especifica o número máximo de réplicas síncronas para cada partição no mapSet. Ele é um atributo opcional e o padrão é 0. Nenhuma outra réplica síncrona é posicionada para uma partição após um domínio alcançar este número de réplicas síncronas para esta partição específica. A inclusão de contêineres que podem suportar esse ObjectGrid pode resultar em um aumento no número de réplicas síncronas se seu valor de maxSyncReplicas ainda não tiver sido atingido. A amostra define o maxSyncReplicas para 1, que significa que o domínio posicionará, no máximo, uma réplica síncrona. Se você iniciar mais de uma instância do servidor de contêineres, haverá somente uma réplica síncrona posicionada em uma das instâncias do servidor de contêineres.

Usando o ObjectGrid

O arquivo `Client.java` no diretório *`installRoot`*`/ObjectGrid/gettingstarted/src/` é o programa cliente que demonstra como se conectar a um servidor de catálogos, obter uma instância de ObjectGrid, e usar a API de ObjectMap.

Da perspectiva de um aplicativo cliente, o uso do WebSphere eXtreme Scale pode ser dividido nas seguintes etapas.

1. Conexão com o serviço de catálogo por meio da obtenção de uma instância de ClientClusterContext.
2. Obtenção de uma instância do ObjectGrid do cliente.
3. Obtenção de uma instância da Sessão.
4. Obtenção de uma instância do ObjectMap.
5. Uso de métodos ObjectMap.

1. Conexão com o serviço de catálogo por meio da obtenção de uma instância de ClientClusterContext

Para se conectar a um servidor de catálogos, use o método connect da API ObjectGridManager. O método connect que é usado por esta amostra somente necessita do terminal do servidor de catálogos no formato de `hostname:port`, como `localhost:2809`. Se a conexão com o servidor de catálogos for bem-sucedida, o método connect retorna ums instância de ClientClusterContext. A instância de ClientClusterContext é necessária para a obtenção do ObjectGrid a partir da API do ObjectGridManager. O trecho de código a seguir demonstra como se conectar a um servidor de catálogos e obter uma instância de ClientClusterContext.

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

2. Obtenção de uma instância do ObjectGrid

Para obter uma instância do ObjectGrid, use o método getObjectGrid da API do ObjectGridManager. O método getObjectGrid necessita de ambos, a instância do ClientClusterContext e o nome da instância do ObjectGrid. A instância do ClientClusterContext é obtida durante a conexão com o servidor de catálogos. O nome do ObjectGrid é "Grid" que é especificado no arquivo `objectgrid.xml`. O trecho de código a seguir demonstra como obter o ObjectGrid chamando o método getObjectGrid da API do ObjectGridManager.

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

3. Obtenção de uma instância da Sessão

É possível obter uma Sessão da instância do ObjectGrid obtida. Uma instância da Sessão é necessária para obter o ObjectMap e executar demarcação de transação. O trecho de código a seguir demonstra como obter a Sessão chamando o método getSession da API do ObjectGrid.

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

4. Obtenção de uma instância do ObjectMap

Após obter uma Sessão, é possível obter o ObjectMap de uma Sessão chamando o método getMap da API da Sessão. Você precisa passar o nome do mapa como parâmetro para o método getMap para obter o ObjectMap. O trecho de código a seguir demonstra como obter ObjectMap chamando o método getMap da API da Sessão.

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

5. Uso de métodos ObjectMap

Após obter um ObjectMap, poderá usar a API do ObjectMap. Lembre-se que ObjectMap é um mapa transacional e necessita de demarcação de transação por meio do uso dos métodos begin e commit da API de Session. Se não existir demarcação de transação explícita, as operações de ObjectMap executam com transações de consolidação automática.

O trecho de código a seguir demonstra como usar a API de ObjectMap com transação de consolidação automática.

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

O trecho de código a seguir demonstra como usar a API de ObjectMap com demarcação de transação explícita.

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

Informações adicionais

Esta amostra demonstra como iniciar o servidor de catálogos e o servidor de contêiner e usar a API do ObjectMap em um ambiente independente. Também é possível usar a API do EntityManager.

Em um ambiente WebSphere Application Server com oWebSphere eXtreme Scale instalado ou ativado, o cenário mais comum é uma topologia conectada à rede. Em

uma topologia conectada à rede, o servidor de catálogos é hospedado no processo do gerenciador de implementação do WebSphere Application Server e cada instância doWebSphere Application Server hospeda um servidor eXtreme Scale automaticamente. Os aplicativos Java™ Platform, Enterprise Edition precisam somente incluir ambos, o arquivo XML descritor do ObjectGrid e o arquivo XML descritor da política de implementação do ObjectGrid, no diretório `META-INF` de qualquer módulo e o ObjectGrid se torna disponível automaticamente. Então, um aplicativo pode se conectar a um servidor de catálogos disponível localmente e obter uma instância do ObjectGrid para uso.

Capítulo 2. Planejando Implementação do Aplicativo

Antes de implementar aplicativos no WebSphere eXtreme Scale, você deve rever os requisitos de hardware e software, implementação de rede e configurações de ajuste, configurações de implementação, e assim por diante. Você também pode usar a verificação operacional para garantir que seu ambiente esteja pronto para ter um aplicativo implementado.

Configurações de Implementação para o eXtreme Scale

O WebSphere eXtreme Scale pode ser implementado em dois tipos de ambientes: local ou distribuído. A configuração necessária depende do tipo do ambiente de implementação.

Ambientes Locais

Ao implementar em um ambiente local, o eXtreme Scale é executado em uma única Java Virtual Machine e não é replicado. Um ambiente local requer um arquivo XML do ObjectGrid ou APIs do ObjectGrid.

Ambientes Distribuídos

Ao implementar em um ambiente distribuído, o eXtreme Scale é executado em um conjunto de Java Virtual Machines. Com essa configuração, poderá utilizar a replicação de dados e criação de partições. Um ambiente distribuído pode ser ainda classificado como uma topologia de implementação dinâmica na qual é possível incluir servidores, conforme necessário. Assim como ocorre com um ambiente local, um arquivo XML do ObjectGrid, ou uma configuração programática equivalente, é necessário em um ambiente distribuído. Além disso, é necessário fornecer um arquivo XML de política de implementação com detalhes de configuração para sua grade de dados de memória do eXtreme Scale.

Requisitos de Hardware e Software

Não é necessário usar um nível específico de hardware ou de sistema operacional para usar o WebSphere eXtreme Scale. As instruções oficiais de suporte do eXtreme Scale são fornecidas on-line na página Requisitos do Sistema.

Para obter uma lista detalhada de opções hardware e software suportados por sistema operacional para o WebSphere eXtreme Scale, consulte a páginaRequisitos do Sistema.

Se houver um conflito entre as informações fornecidas no centro de informações e as informações sobre a página Requisitos do Sistema, as informações no Web site tem precedência. As informações de pré-requisito no centro de informações são fornecidas apenas como uma conveniência.

Requisitos de Hardware

O WebSphere eXtreme Scale não requer um nível específico de hardware. Os requisitos de hardware dependem do hardware suportado para a instalação do Java Platform, Standard Edition que é utilizado para executar o WebSphere eXtreme Scale. Se você estiver usando o eXtreme Scale com oWebSphere

Application Server ou outra implementação do Java Platform, Enterprise Edition, os requisitos de hardware dessas plataformas são suficientes para oWebSphere eXtreme Scale.

Requisitos do Sistema Operacional

O eXtreme Scale não requer um nível de sistema operacional específico. Cada implementação de Java SE e Java EE necessita de diferentes níveis de sistema operacional ou correções para problemas que são descobertos durante o teste da implementação de Java. Os níveis que as implementações necessitam são suficientes para oeXtreme Scale.

Requisitos do WebSphere Application Server

Clientes e servidores do eXtreme Scale em execução em um ambiente distribuído e ObjectGrids de memória local são suportados no WebSphere Application Server Versão 6.0.2 e superior.

Nota: Para usar o provedor de cache dinâmico, o seu sistema deve satisfazer a um dos seguintes requisitos mínimos:

- WebSphere Application Server Versão 6.1.0.25 ou mais recente + Interim Fix PK85622
- WebSphere Application Server Versão 7.0.0.3 ou mais recente + Interim Fix PK85622

Consulte as Correções recomendadas para o WebSphere Application Server para obter informações adicionais.

Outros Requisitos do Servidor de Aplicativos

Outras implementações de Java EE podem usar o tempo de execução do eXtreme Scale como uma instância local ou como um cliente para servidores eXtreme Scale. Para implementar o Java SE, você deve usar a Versão 1.4.2 ou mais recente.

Ajuste do desempenho

Para melhorar o desempenho, considere determinados itens, como sistema operacional e ajuste de rede, planejamento de portas de rede, ajuste do Java Virtual Machine (JVM) para configurações do WebSphere eXtreme Scale, configuração de failover e outros tópicos de ajuste.

Planejamento para Portas de Rede

O WebSphere eXtreme Scale é um cache distribuído que necessita de portas de abertura para se comunicar com o Object Request Broker (ORB) e a pilha do Protocolo de Controle de Transmissões (TCP) entre Java Virtual Machines e outras máquinas. Você deve planejar e controlar suas portas, especialmente em um ambiente de firewall. Por exemplo, ao usar serviço de catálogo e contêineres para abrir várias portas.

Grade de Serviço de Catálogo

Uma grade de serviço de catálogo requer o seguinte:

1. peerPort para que o gerenciador de alta disponibilidade (HA) se comunicar entre servidores de catálogo peer em uma pilha TCP

2. clientPort para que os servidores de catálogo acessem dados de serviço de catálogo
3. JMXServicePort para especificar qual porta o serviço JMX deve usar
4. Porta listener ORB para que contêineres e clientes se comuniquem com o serviço de catálogo através de ORB

As portas previamente listadas podem ser especificadas pelo comando startOgServer com a opção em modo independente, como mostra o exemplo a seguir:

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

No ambiente WebSphere Application Server, as portas na lista anterior são especificadas nas propriedades customizadas da célula do WebSphere com a chave catalog.services.cluster como:

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

Os servidores de contêineres do WebSphere eXtreme Scale também precisam de várias portas para operar. Por padrão, o servidor de contêineres do eXtreme Scale gera sus porta do gerenciador HA e porta listener ORB automaticamente com portas dinâmicas. No ambiente de firewall, como é vantajoso planejar e controlar portas, as opções são fornecidas para iniciar os servidores de contêineres do eXtreme Scale com porta HAManager e porta listener ORB especificadas com uma opção no comando startOgServer, como mostra o exemplo a seguir:

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

O planejamento adequado do controle da porta é um benefício, mas existe uma dificuldade inerente no planejamento e gerenciamento dessas portas quando centenas de Java Virtual Machines são iniciadas em uma máquina. Qualquer conflito de porta causará uma falha da inicialização do servidor.

Quando a segurança está ativada, uma porta Secure Socket Layer (SSL) é uma inclusão necessária às portas listadas anteriormente. Usar `Dcom.ibm.CSI.SSLPort=<sslPort>` como um argumento -jvmArgs configura a porta SSL para `<sslPort>`.

Ajuste de Sistemas Operacionais e Rede

O ajuste da rede pode reduzir o atraso da pilha do Protocolo de Controle de Transmissões (TCP) através da alteração das configurações de keep alive da conexão e aumentar o rendimento pela mudança dos buffers TCP.

Sistemas Operacionais

Um sistema Windows® precisa do mínimo de ajuste enquanto que o sistema Solaris precisa do máximo de ajuste. As informações a seguir referem-se a cada sistema especificado e podem aumentar o desempenho do WebSphere eXtreme Scale. Você deve ajustar de acordo com sua carga de rede e do aplicativo.

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
```

Propriedades do ORB e Configurações do Descritor de Arquivo

As considerações de ajuste incluem propriedades do Object Request Broker (ORB) e configurações do descritor de arquivo.

Propriedades do ORB

O ORB é usado pelo WebSphere eXtreme Scale para se comunicar através de uma pilha TCP. O arquivo `orb.properties` necessário está no diretório `java/jre/lib`. Para um carga pesada de objetos grandes, ative a fragmentação de ORB especificando as seguintes configurações:

```
com.ibm.CORBA.FragmentSize=<tamanho correto>
```

Evite o crescimento de ThreadPool especificando as seguintes configurações:

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

Configure os tempos limite para evitar excesso de encadeamentos em uma situação anormal especificando as seguintes configurações:

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

Descritor de Arquivo

Sistemas UNIX® e Linux® possuem um limite para a quantidade de arquivos abertos permitidos por processo. O sistema operacional especifica a quantidade de

arquivos abertos permitidos. Se este valor for configurado com um valor muito baixo, um erro de alocação de memória ocorrerá no AIX, e muitos arquivos abertos serão registrados no log.

Na janela de terminal do sistema UNIX, configure este valor para um valor mais alto que o valor do sistema padrão. Para máquinas SMP maiores com clones, configure para ilimitado.

Para configurações AIX, defina este valor como `-1` (ilimitado) com o comando `ulimit -n -1`.

Para configurações Solaris, defina este valor como `16384` com o comando `ulimit -n 16384`.

Para exibir o valor atual, use o comando `ulimit –a`.

Ajuste da JVM para o WebSphere eXtreme Scale

Esta página discute algumas questões específicas sobre o ajuste da JVM (Java Virtual Machine) para WebSphere eXtreme Scale. Se tiver problemas de desempenho, entre em contato com o suporte da IBM® support. Esta página será atualizada com as informações de ajuste adicionais conforme elas são disponibilizadas.

Na maioria dos casos, o WebSphere eXtreme Scale requer pouca ou nenhuma configuração especial da JVM. Se você tiver uma grande quantidade de objetos sendo armazenados em um WebSphere eXtreme Scale, ajuste o tamanho de heap para um nível apropriado para evitar a execução sem memória.

Plataformas

Testes de desemenho ocorridos primariamente nas caixas AIX (32 processadores), Linux (4 processadores), Windows (8 processadores) e AZUL (384 processadores). Com sistemas AZUL e caixas AIX high-end, cenários fortemente multiencadeados podem ser armazenados e os pontos de contenção podem ser identificados e corrigidos. Os sistemas AZUL também eliminam a coleta de lixo, o que pode contaminar grandemente os ajustes executados durante o teste.

Requisitos do Object Request Broker (ORB)

O IBM SDK inclui uma implementação IBM ORB que foi testada com o WebSphere Application Server e o WebSphere eXtreme Scale. Para facilitar o processo de suporte, use um JVM fornecido pela IBM. Outras implementações de JVM utilizam um ORB diferente. O IBM ORB é fornecido apenas pronto para uso com as Java virtual machines fornecidas pela IBM. O WebSphere eXtreme Scale requer um ORB em funcionamento para operar. É possível utilizar o WebSphere eXtreme Scale com ORBs de terceiros, mas se ocorrer um problema no ORB, será necessário entrar em contato com o fornecedor do ORB para obter suporte. A implementação do IBM ORB é compatível com Java virtual machines de terceiros e pode ser substituído, se necessário.

Coleta de Lixo

O WebSphere eXtreme Scale cria objetos temporários associados a cada transação, como pedido e reposta e sequência de log. Como esses objetos afetam a eficiência da coleta de lixo, o ajuste da coleta de lixo é fundamental.

Para a máquina virtual IBM para Java, use o coletor optavgpause para cenários com alta taxa de atualização (100% das transações modificam entradas). O coletor gencon funciona bem melhor que o coletor optavgpause nos cenários em que os dados são atualizados relativamente com pouca frequência (10% do tempo ou menos). Experimente ambos os coletores para ver qual funciona melhor no seu cenário. Em caso de algum problema de desempenho, ative a coleta de lixo detalhada para verificar a porcentagem de tempo que está sendo consumida pela coleta. Ocorreram cenários em que 80% do tempo foi gasto na coleta de lixo até que um que o ajuste resolvesse o problema.

Para obter mais informações sobre a coleta de lixo, consulte Ajustando a Máquina Virtual IBM para Java.

Desempenho da JVM

O WebSphere eXtreme Scale pode executar em diferentes versões de J2SE (Java 2 Platform, Standard Edition). O WebSphere eXtreme Scale Versão 6.1 suporta o J2SE Versão 1.4.2 e superior. Para produtividade e desempenho de desenvolvedor aprimorados, utilize o J2SE 5 ou mais recente para obter vantagem das anotações e da coleta de lixo aprimorada. O WebSphere eXtreme Scale funciona em Java virtual machines de 32 ou 64 bits.

O WebSphere eXtreme Scale é testado com um subconjunto das máquinas virtuais disponíveis, todavia, a lista suportada não é exclusiva. É possível executar WebSphere eXtreme Scale em qualquer Versão 1.4.2 ou posterior mas, se for identificado um problema na JVM, você deve entrar em contato com o fornecedor da JVM para obter suporte. Se possível, use a JVM a partir do tempo de execução do WebSphere em qualquer plataforma que o WebSphere Application Server suporta.

Para a maioria dos cenários nos quais o WebSphere eXtreme Scale é usado, o Java Platform Standard Edition 6 da JVM funciona melhor que o Edition 5 ou 1.4. O Java 2 Platform, Standard Edition Versão 1.4 tem um fraco desempenho, principalmente em cenários que usam o coletor gencon. A Java Platform Standard Edition 5 executa bem, as a Java Platform, Standard Edition 6 executa melhor.

Heaps Grandes

Quando o aplicativo precisa gerenciar uma grande quantidade de dados para cada partição, a coleta de lixo pode ser um fator. Na maioria das vezes, o desempenho de leitura é bom mesmo com heaps muito grandes (20 GB ou mais) quando um coletor geracional é utilizado. Contudo, depois que o heap de estabilidade é preenchido, ocorre uma pausa proporcional ao tamanho do heap ativo e ao número de processadores na máquina. Essa pausa pode ser grande em máquinas menores com grandes heaps.

O WebSphere eXtreme Scale suporta o WebSphere Real Time Java. Com o WebSphere Real Time Java, a resposta de processamento de transação do WebSphere eXtreme Scale é mais consistente e previsível e o impacto da coleta de lixo e do planejamento do encadeamento é enormemente minimizado. O impacto é reduzido ao nível no qual o tempo de desvio de resposta padrão é inferior a 10% da Java regular.

Consulte “Usando o WebSphere Real Time” na página 212 para obter informações adicionais.

Contagem de Encadeamentos

A contagem de encadeamentos depende de poucos fatores. Há um limite de quantos encadeamentos um único shard pode gerenciar. Um shard é uma instância de uma partição e pode ser primário ou de réplica. Com mais shards para cada JVM, poderá haver mais encadeamentos, com cada shard fornecendo mais caminhos simultâneos para os dados. Embora cada shard é simultâneo o máximo possível, essa simultaneidade ainda é limitada.

Configurando a Detecção de Failover

É possível configurar a quantia de tempo entre as verificações do sistema para servidores com falha. Esse valor é chamado de intervalo de pulsação.

Por Que e Quando Desempenhar Esta Tarefa

A configuração de failover varia dependendo do tipo de ambiente que você está usando. Se você estiver utilizando um ambiente independente, é possível configurar o failover com a linha de comandos. Se você estiver usando um ambiente do WebSphere Application Server Network Deployment, é necessário configurar o failover no console administrativo do WebSphere Application Server Network Deployment.

- Configurar o failover para ambientes independentes.

  É possível configurar os intervalos de pulsação na linha de comandos usando o parâmetro **-heartbeat** no arquivo de script startOgServer.bat | startOgServer.sh. Configure esse parâmetro para um dos seguintes valores:

  Tabela 1. Intervalos de Pulsações

Valor	Ação	Descrição
0	Típica (padrão)	Failovers são tipicamente detectados em 30 segundos.
-1	Agressiva	Failovers são tipicamente detectados em 5 segundos.
1	Moderada	Failovers são tipicamente detectado em 180 segundos.

  Um intervalo de pulsação agressivo pode ser útil quando os processos e a rede estão estáveis. Se a rede ou os processos não são configurados de maneira ideal, as pulsações podem ser perdidas, o que pode resultar em uma falsa detecção de falhas.

- Configurar o failover para ambientes do WebSphere Application Server.

  O WebSphere Application Server Network Deployment Versão 6.0.2 e superior pode ser configurado para permitir que o WebSphere eXtreme Scale execute failover muito rapidamente. O tempo de failover padrão para falhas ″hard″ é de aproximadamente 200 segundos. Uma falha ″hard″ é um travamento de computador físico ou de servidor, uma desconexão de cabo de rede ou um erro do sistema operacional. As falhas causadas por travamentos de processo ou por falhas de software normalmente executam failover em menos de um segundo. A detecção de falha para falhas ″soft″ ocorre quando os soquetes de rede a partir de um processo inativo são fechados automaticamente pelo sistema operacional para o servidor que hospeda o processo.

  Configuração de pulsação do grupo principal

  O WebSphere eXtreme Scale que executa um processo do WebSphere Application Server herda as características de failover a partir das configurações de grupo principal do servidor de aplicativos. As seguintes seções descrevem como definir as configurações de pulsação do grupo principal para versões diferentes do WebSphere Application Server Network Deployment:

– **Atualize as configurações do grupo principal para o WebSphere Application Server Network Deployment Versão 6.x e 7.x:**

  O intervalo de pulsação pode ser especificado em segundos nas versões do WebSphere Application Server da Versão 6.0 a Versão 6.1.0.12, ou em milissegundos iniciando na Versão 6.1.0.13. Também é necessário especificar o número de pulsações perdidas. Esse valor indica quantas pulsações podem estar ausentes antes que um Java Virtual Machine (JVM) equivalente seja considerado uma falha. O tempo de detecção de falha ″hard″ é, aproximadamente, o produto do intervalo de pulsações e o número de pulsações ausentes.

  Essas propriedades são especificadas usando as propriedades customizadas no grupo principal usando o console administrativo do WebSphere. Consulte Propriedades customizadas do grupo principal para obter detalhes da configuração. Estas propriedades devem ser especificadas para todos os grupos principais utilizados pelo aplicativo:

  - O intervalo de pulsação é especificado usando a propriedade customizada IBM_CS_FD_PERIOD_SEC para segundos ou a propriedade customizada IBM_CS_FD_PERIOD_MILLIS para milissegundos (requer o V6.1.0.13 ou superior).
  - O número de pulsações ausentes é especificado usando a propriedade customizada IBM_CS_FD_CONSECUTIVE_MISSED.

  O valor padrão para a propriedade IBM_CS_FD_PERIOD_SEC é 20 e para a propriedade IBM_CS_FD_CONSECUTIVE_MISSED é 10. Se a propriedade IBM_CS_FD_PERIOD_MILLIS for especificada, ela substituirá qualquer conjunto de propriedades customizadas IBM_CS_FD_PERIOD_SEC. Os valores destas propriedades são valores de número inteiro positivo.

  Use as seguintes configurações para alcançar um tempo de detecção de falha de 1500 milissegundos para os servidores WebSphere Application Server Network Deployment Versão 6.x:

  - Configure `IBM_CS_FD_PERIOD_MILLIS = 750` (WebSphere Application Server Network Deployment V6.1.0.13 e superior)
  - Configure `IBM_CS_FD_CONSECUTIVE_MISSED = 2`

– **Atualize as configurações do grupo principal para o WebSphere Application Server Network Deployment Versão 7.0:**

  O WebSphere Application Server Network Deployment Versão 7.0 fornece duas configurações de grupo principal que podem ser ajustadas para aumentar ou diminuir a detecção de failover:

  - **Período de transmissão de pulsação.** O padrão é 30000 milissegundos.
  - **Período de tempo limite de pulsação.** O padrão é 180000 milissegundos.

  Para obter mais detalhes sobre como alterar essas configurações, consulte o Centro de Informações do WebSphere Application Server Network Deployment: Configurações de Falha e Detecção de Descoberta.

  Use as seguintes configurações para alcançar um tempo de detecção de falha de 1500 milissegundos para os servidores WebSphere Application Server Network Deployment Versão 7:

  - Configure o período de transmissão de pulsação para 750 milissegundos.
  - Configure o tempo limite da pulsação para 1500 milissegundos.

O que Fazer Depois

Quando estas configurações são modificadas para fornecer tempos de failover curtos, há alguns problemas de ajuste de sistema a considerar. Primeiro, Java não é

um ambiente em tempo real. É possível que os encadeamentos sejam atrasados se a JVM estiver experimentando longos tempos de coleta de lixo. Os encadeamentos também podem ser atrasados se a máquina que hospeda o JVM estiver sobrecarregada (devido ao próprio JVM ou outros processos que são executados na máquina). Se os encadeamentos forem atrasados, as pulsações talvez não sejam enviadas a tempo. No pior dos casos, elas podem ser atrasadas pelo tempo necessário de failover. Se os encadeamentos forem atrasados, ocorrerão falsas detecções de falhas. O sistema deve ser ativado e dimensionado para garantir que falsas detecções de falhas não aconteçam na produção. O teste de carregamento adequado é a melhor maneira de garantir isto.

Nota: A versão atual do eXtreme Scale suporta o WebSphere Real Time.

Serviço de Catálogo de Alta Disponibilidade

Um serviço de catálogo é a grade de servidores de catálogo que você está usando, que retém as informações de topologia para todos os contêineres no seu ambiente do eXtreme Scale. O serviço de catálogo controla o equilíbrio e o roteamento de todos os clientes. Para implementar o eXtreme Scale como um espaço de processamento de banco de dados de memória, poderá armazenar em cluster o serviço de catálogo em uma grade para alta disponibilidade.

Componentes do Serviço de Catálogo

Quando múltiplos servidores de catálogo iniciam, um dos servidores é eleito como o servidor de catálogo principal que aceita pulsações do Internet Inter-ORB Protocol (IIOP) e manipula as alterações dos dados do sistema em resposta a qualquer serviço de catálogo ou as alterações de contêiner.

Quando os clientes entram em contato com qualquer um dos servidores de catálogo, a tabela de roteamento para a grade do servidor de catálogos é propagada para os clientes através do contexto de serviço Common Object Request Broker Architecture (CORBA).

Configure pelo menos três servidores de catálogos. Se a sua configuração tem zonas, é possível configurar um servidor de catálogos por zona.

Quando um servidor e contêiner do eXtreme Scale entra em contato com um dos servidores de catálogos, a tabela de roteamento para a grade do servidor de catálogos também é propagada para o servidor e contêiner do eXtreme Scale através do contexto de serviço CORBA. Além disso, se o servidor de catálogos contactado atualmente não for o servidor de catálogos principal, o pedido é automaticamente roteado para o servidor de catálogos principal atual e a tabela de roteamento para o servidor de catálogos é atualizada.

Nota: Uma grade do servidor de catálogos e a grade do servidor de contêineres são muito diferentes. A grade do servidor de catálogos é para a alta disponibilidade dos seus dados do sistema. A grade do contêiner é para a alta disponibilidade de dados, escalabilidade e gerenciamento de carga de trabalho. Portanto, existem duas diferentes tabelas de roteamento: a tabela de roteamento para a grade do servidor de catálogos e a tabela de roteamento para os shards da grade do servidor.

As responsabilidades do catálogo são divididas em uma série de serviços. O gerenciador do grupo principal executa agrupamento peer para monitoramento de funcionamento, o serviço de posicionamento executa alocação, o serviço de

administração fornece acesso à administração e o serviço de local gerencia a localidade.

Implementação da Grade de Catálogo

Gerenciador de grupo principal

O serviço de catálogo usa o gerenciador de alta disponibilidade (gerenciador HA) para agrupar processos para o monitoramento de disponibilidade. Cada agrupamento dos processos é um grupo principal. Com o eXtreme Scale, o gerenciador do grupo principal agrupa dinamicamente os processos. Estes processos são mantidos pequenos para permitir a escalabilidade. Cada grupo principal elege um líder que possui a responsabilidade adicional de enviar o status para o gerenciador do grupo principal quando membros individuais falham. O mesmo mecanismo de status é usado para descobrir quando todos os membros de um grupo falham, o que causa a falha da comunicação com o líder.

O gerenciador do grupo principal é um serviço completamente automático responsável pela organização de contêineres em pequenos grupos de servidores que são então automaticamente associados de maneira livre para criar um ObjectGrid. Quando um contêiner entra em contato com o serviço de catálogo pela primeira vez, ele aguarda para ser designado a um grupo novo ou existente de vários. O eXtreme Scale consiste de diversos destes grupos e este agrupamento é um importante ativador de escalabilidade. Cada grupo é um grupo do Java Virtual Machines que usa a pulsação para monitorar a disponibilidade dos outros grupos. Um destes membros do grupo é eleito o líder e possui uma responsabilidade adicional de retransmitir informações de disponibilidade para o serviço de catálogo para permitir reação através de realocação e encaminhamento de rotas.

Serviço de disposição

O serviço de catálogo gerencia o posicionamento de shards por todo o conjunto de contêineres disponíveis. O serviço de disposição é responsável pela manutenção de um equilíbrio de recursos entre recursos físicos. O serviço de disposição é responsável pela alocação de shards individuais em seu contêiner de host. Ele executa como um serviço eleito ″um-entre-N″ na grade de forma que sempre exista exatamente uma instância do serviço em execução. Se tal instância falhar, outro processo é então eleito e assume. Para redundância, o estado do serviço de catálogo é replicado entre todos os servidores que estão hospedando o serviço de catálogo.

Administração

O serviço de catálogo também é o ponto de entrada lógico para administração do sistema. O serviço de catálogo hospeda um Managed Bean (MBean) e fornece as URLs do Java Management Extensions (JMX) para qualquer um dos servidores que o serviço está gerenciando.

Serviço de local

O serviço de local atua como o ponto de contato para ambos os clientes que estão procurando contêineres que hospedam o aplicativo que procuram, bem como os contêineres que estão registrando aplicativos hospedados com o serviço de disposição. O serviço de local é executado em todos os membros da grade para efetuar o scale out desta função.

O serviço de catálogo hospeda lógica que é tipicamente inativa durante estados estáveis. Como resultado, o serviço de catálogo influencia a escalabilidade minimamente. O serviço é baseado em centenas de serviços de contêineres que se tornam disponíveis simultaneamente. Para disponibilidade, configure o serviço de catálogo numa grade.

Planejamento

Após o início de uma grade de catálogo, os membros da grade se ligam. Cuidadosamente, planeje a topologia da sua grade de catálogo porque você não pode modificar a configuração do catálogo no tempo de execução. Disperse a grade o mais diversamente possível para evitar erros.

Iniciando uma grade do servidor de catálogo

Conectando os contêineres integrados no eXtreme Scale no WebSphere Application Server para uma grade de catálogo independente

É possível configurar os contêineres doeXtreme Scale que estão integrados em um ambiente do WebSphere Application Server para conectar-se a uma grade de catálogo independente. Use a mesma propriedade para conectar o servidor de aplicativos à grade de catálogo. Entretanto, a propriedade não gerencia o ciclo de vida do catálogo com os servidores. Em vez disso, a propriedade permite que os contêineres localizem a grade de catálogo remota. Para configurar a propriedade, consulte.

Nota: Conflito de nome do servidor: Como esta propriedade é usada para iniciar o servidor de catálogos do eXtreme Scale, assim como para se conectar a ele, os servidores de catálogo não devem ter o mesmo nome de nenhum servidor do WebSphere Application Server.

Consulte as informações sobre os quoruns do servidor de catálogos no *Visão Geral do Produto* para obter mais informações.

Lista de Verificação Operacional

Use a lista de verificação operacional para preparar o ambiente para implementar o WebSphere eXtreme Scale.

Tabela 2. Lista de Verificação Operacional

Item de Lista de Verificação	Para obter informações adicionais

Se você estiver usando o AIX, ajuste as seguintes configurações do sistema operacional:
 	
TCP_KEEPINTVL
A configuração TCP_KEEPINTVL faz parte de um protocolo de soquete keep-alive que permite a detecção de uma interrupção de rede. A propriedade especifica o intervalo entre os pacotes que são enviados para validar a conexão. Ao usar o WebSphere eXtreme Scale, configure o valor para `10`. Para verificar a configuração atual, execute o seguinte comando:
`# no –o tcp_keepintvl`
Para alterar a configuração atual, execute o seguinte comando:
`# no –o tcp_keepintvl=10`
A configuração TCP_KEEPINTVL está em unidades de meio segundo.
 	
TCP_KEEPINIT
A configuração TCP_KEEPINIT faz parte de um protocolo de soquete keep-alive que permite a detecção de uma interrupção de rede. A propriedade especifica o valor de tempo limite inicial para a conexão TCP. Ao usar o WebSphere eXtreme Scale, configure o valor para `40`. Para verificar a configuração atual, execute os seguintes comandos:
`# no –o tcp_keepinit`
Para alterar a configuração atual, execute o seguinte comando:
`# no –o tcp_keepinit=40`
A configuração TCP_KEEPINIT está em unidades de meio segundo.
	• Para obter informações de ajusta do AIX, consulte Ajustando Sistemas AIX.

Atualize o arquivo `orb.properties` para modificar o comportamento de transporte da grade. O arquivo `orb.properties` está no diretório `java/jre/lib`.	"Arquivo de Propriedades ORB" na página 197

Tabela 2. Lista de Verificação Operacional (continuação)

Item de Lista de Verificação	Para obter informações adicionais
Use parâmetros no script startOgServer. Em particular, use os seguintes parâmetros: • Defina as configurações de heap com o parâmetro **-jvmArgs**. • Defina o caminho de classe e as propriedades do aplicativo com o parâmetro **-jvmArgs**. • Defina os parâmetros **-jvmArgs** para configurar o monitoramento de agente. **Configurações de Porta** O WebSphere eXtreme Scale precisa abrir portas para comunicações para alguns transportes. Essas portas são todas definidas dinamicamente. Porém, se um firewall estiver em uso entre os contêineres, será necessário especificar as portas. Use as seguintes informações sobre as portas: **Porta Listener** Você pode usar o argumento **-listenerPort** para especificar a porta que é usada para comunicação entre processos. **Porta do grupo principal** Você pode usar o argumento **-haManagerPort** para especificar a porta que é usada para detecção de falha. Observe que os grupos principais não precisam se comunicar entre as zonas, portanto, pode não ser necessário configurar essa porta se o firewall estiver aberto para todos os membros de uma única zona. **Porta de serviço JMX** Você pode usar o argumento **-JMXServicePort** para especificar a porta que o serviço JMX deve usar. **Porta SSL** Transmitir `-Dcom.ibm.CSI.SSLPort=1234` como um argumento **-jvmArgs** configura a porta SSL para `1234`. A porta SSL é a porta protegida equivalente à porta listener. **Porta do cliente** Usada apenas no serviço de catálogo. Você pode especificar esse valor com o argumento **-catalogServiceEndPoints**. O formato do valor desse parâmetro está no formato: `serverName:hostName:clientPort:peerPort`	"Script startOgServer" na página 223
Verifique se as configurações de segurança estão definidas corretamente: • Transporte (SSL) • Aplicativo (Autenticação e Autorização) Para verificar as configurações de segurança, é possível tentar usar um cliente malicioso para conectar-se com a sua configuração. Por exemplo, quando a configuração SSL necessária for definida, um cliente que possuir a configuração TCP_IP ou um cliente com o truststore incorreto não conseguirá se conectar ao servidor. Quando a autenticação é necessária, um cliente com nenhuma credencial, como um ID de usuário e senha, não conseguirá se conectar ao servidor. Quando a autorização é forçada, um cliente com nenhuma autorização de acesso não conseguira acessar os recursos do servidor.	Capítulo 9, "Protegendo o Ambiente de Implementação", na página 285
Escolha como você monitorará seu ambiente. • xsAdmin – As portas JMX dos servidores de catálogo precisam estar visíveis para a ferramenta XSAdmin. As portas do contêiner também precisam estar acessíveis para alguns comandos que reúnem informações a partir dos contêineres. • É possível escolher entre as seguintes ferramentas de monitoramento do fornecedor: – Tivoli Enterprise Monitoring Agent – CA Wily Introscope – Hyperic HQ	• "Utilizando o Utilitário de Amostra xsAdmin" na página 266 • "Segurança do Java Management Extensions (JMX)" na página 296 • "Monitorando com o IBM Tivoli Enterprise Monitoring Agent para WebSphere eXtreme Scale" na página 268 • "Monitorando o eXtreme Scale com o Hyperic HQ" na página 277 • "Monitorando Aplicativos eXtreme Scale com o CA Wily Introscope" na página 274

Capítulo 3. Planejamento de Capacidade

Se você tiver um tamanho de conjunto de dados inicial e um tamanho de conjunto de dados projetado, é possível planejar a capacidade necessária para executar o WebSphere eXtreme Scale. Embora esse planejamento ajude a implementar o eXtreme Scale eficientemente para futuras alterações, ele permite maximizar a elasticidade do eXtreme Scale o que não aconteceria em um cenário diferente, como um banco de dados de memória ou outro tipo de banco de dados.

Grades, Partições e Shards

Uma grade distribuída do eXtreme Scale é dividida em partições. Uma partição retém um subconjunto exclusivo de dados. Uma partição é constituída de um ou mais shards: um shard primário e shards réplicas. Não é necessário ter shards réplicas em uma partição, mas os shards réplicas fornecem alta disponibilidade.Independente se sua implementação for um espaço de processamento de grade de dados ou de banco de dados de memória independente, o acesso aos dados no eXtreme Scale depende fortemente dos conceitos dos shards.

Os dados para uma partição são armazenados no tempo de execução em um conjunto de shards. Este conjunto de shards inclui um shard principal e, possivelmente, um ou mais shards de réplica. Um shard é a menor unidade que o eXtreme Scale pode incluir ou remover de uma Java Virtual Machine.

Existem duas estratégias de posicionamento: FIXED_PARTITIONS (padrão) e PER_CONTAINER. A seguinte abordagem foca o uso da estratégia FIXED_PARTITIONS.

Número de Shards

Se o seu ambiente incluía dez partições que continham um milhão de objetos sem réplica, então, existiriam 10 shards com cada um armazenando 100.000 objetos. Se você incluir uma réplica neste cenário, então, existe um shard extra em cada partição. Neste caso, existem 20 shards - dez shard primários e dez shards de réplica. Novamente, cada um destes shards armazenaria 100.000 objetos. Cada partição consiste de um shard primário e uma ou mais (N) shards de réplica. Determinar a contagem de shards ideal é crítica. Se você configurar um pequeno número de shards, os dados não são distribuídos igualmente entre os shards, resultando em erros de falta de memória e problemas de sobrecarga do processador. É necessário ter pelo menos dez shards para cada JVM à medida que escala. Quando você está implementando a grade inicialmente, poderia potencialmente utilizar um grande número de partições.

Número de Shards por JVM

Cenário: pequeno número de shards para cada JVM

Os dados são incluídos e removidos de uma JVM utilizando unidades de shard. Os shards nunca são divididos em partes. Se existirem 10 GB de dados e 20 shards para conterem estes dados, cada shard conterá 500 MB de dados em média. Se nove Java Virtual Machines hospedam a grade, então, em média, cada JVM possui

dois shards. Como 20 não é igualmente divisível por 9, algumas Java Virtual Machines possuem três shards, na seguinte distribuição:

- 7 Java Virtual Machines com 2 shards
- 2 Java Virtual Machines com 3 shards

Como cada shard contém 500 MB de dados, a distribuição de dados é desigual. As sete Java Virtual Machines com dois shards contêm cada uma 1 GB de dados. As duas Java Virtual Machines com três shards possuem 50% mais dados ou 1.5 GB, o que é uma carga de memória muito maior. Como estas duas Java Virtual Machines estão hospedando três shards, elas também recebem 50% mais pedidos para seus dados. Como resultado, um pequeno número de shards para cada JVM causa desequilíbrio. Para aumentar o desempenho, aumente o número de shards para cada JVM.

Cenário: número aumentado de shards por JVM

Neste cenário, considere um número muito maior de shards. Neste cenário, há 101 shards com 9 Java Virtual Machines hospedando 10 GB de dados. Neste caso, cada shard contém 99 MB de dados. A Java Virtual Machines possui a seguinte distribuição de shards:

- 7 Java Virtual Machines com 11 shards
- 2 Java Virtual Machines com 12 shards

As duas Java Virtual Machines com 12 shards agora possuem apenas mais 99 MB de dados do que os outros shards, o que é uma diferença de 9%. Este cenário é muito mais igualmente distribuído do que a diferença de 50% no cenários com um pequeno número de shards. A partir de uma perspectiva de uso do processador, existe apenas 9% mais de trabalho para as duas Java Virtual Machines com os 12 shards comparado às sete Java Virtual Machines que possuem 11 shards. Ao aumentar o número de shards em cada JVM, o uso dos dados e do processador é distribuído de uma maneira proporcional e equilibrada.

Quando você está criando seu sistema, utilize 10 shards para cada JVM em seu cenário com dimensionamento máximo ou quando o sistema está executando seu número máximo de Java Virtual Machines em seu horizonte de planejamento.

Dimensionamento de Memória e Cálculo de Contagem de Partições

É possível calcular a quantidade de memória e de partições que são necessárias para sua configuração.

O WebSphere eXtreme Scale armazena dados no espaço de endereço das Java Virtual Machines (JVM). Cada JVM fornece espaço no processador para criar, recuperar, atualizar e excluir chamadas para os dados que estão armazenados na JVM. Além disso, cada JVM fornece espaço de memória para entradas de dados e réplicas. Objetos Java variam de tamanho, assim você deve medir para fazer uma estimativa de quanta memória você precisa.

Para dimensionar a memória necessária, carregue os dados do seu aplicativo em uma única JVM. Quando o uso do heap alcança 60%, observe o número de objetos que são utilizados. Este número é a contagem máxima recomendada de objetos para cada uma de suas Java Virtual Machines. Para obter o dimensionamento mais exato, utilize dados realistas e inclua quaisquer índices definidos em seu dimensionamento porque os índices também consomem memória. A melhor forma de dimensionar o uso da memória é executar a saída verbosegc de coleta de lixo

pois esta saída fornece os números após a coleta de lixo. É possível consultar o uso do heap em qualquer ponto específico através de MBeans ou programaticamente, mas tais consultas fornecem apenas uma captura instantânea atual do heap, o que pode incluir lixo não coletado, sendo que, desta forma, o método não é uma indicação precisa da memória consumida.

Escalando a Configuração

Quantidade de shards por partição (numShardsPerPartition value)

Para calcular a quantidade de shards por partição, ou o valor de numShardsPerPartition, inclua 1 para o shard primário mais a quantidade total de shards de réplica que desejar.

```
numShardsPerPartition = 1 + total_number_of_replicas
```

Quantidade de Java Virtual Machines (valor minNumJVMs)

Para escalar sua configuração, primeiro, decida o número máximo de objetos necessários a serem armazenados no total. Para determinar a quantidade de Java Virtual Machines que precisa, use a seguinte fórmula:

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

Arredonde este valor para cima para o valor de número inteiro mais próximo.

. arredonde para o número inteiro mais próximo

Quantidade de shards (valor numShards)

No tamanho de crescimento final, devem ser utilizados 10 shards para cada JVM. Conforme descrito antes, cada JVM possui um shard principal e (N-1) shards para as réplicas, ou neste caso, 9 réplicas. Como você já possui uma quantidade de Java Virtual Machines para armazenar os dados, você pode multiplicar a quantidade de Java Virtual Machines por 10 para determinar a quantidade de shards:

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

Número de partições

Se uma partição tiver um shard primário e um shard de réplica, então a partição possui dois shards (principal e de réplica). O número de partições é a contagem de shards dividido por 2, arredondado para o número primo mais próximo. Se a partição possui um primário e duas réplicas, então, o número de partições é a contagem de shards dividida por 3, arredondada para o número primo mais próximo.

```
numPartitions = numShards / numShardsPerPartition
```

Exemplo de Escala

Neste exemplo, o número de entrada inicia em 250 milhões. A cada ano, o número de entradas aumenta em cerca de 14%. Após 7 anos, o número total de entrada será de 500 milhões, portanto, planeje sua capacidade de acordo. Para alta disponibilidade, uma única réplica é necessária. Com uma réplica, o número de entradas duplica, ou seja, é de 1 bilhão de réplicas. Como um teste, 2 milhões de entradas podem ser armazenadas em cada JVM. Utilizando os cálculos neste cenário, a seguinte configuração é necessária:

- 500 Java Virtual Machines para armazenar o número final de entradas.

- 5000 shards, calculados ao multiplicar 500 Java Virtual Machines por 10.
- 2500 partições, ou 2503 como o próximo número primo mais alto, calculados ao obter 5000 shards, divididos por dois para shards primários e de réplica.

Iniciando a Configuração

Com base nos cálculos anteriores, você deveria iniciar com 250 Java Virtual Machines e crescer até 500 Java Virtual Machines ao longo de 5 anos, o que lhe permite gerenciar o crescimento incremental até chegar na quantidade final de entradas.

Nesta configuração, cerca de 200.000 entradas são armazenadas por partição (500 milhões de entradas divididas por 2503 partições). É necessário configurar o parâmetro numberOfBuckets no mapa que contém as entradas para o número primo mais alto e mais próximo, neste exemplo 70887, que mantém a proporção por volta de 3.

Quando o número máximo de Java Virtual Machines é atingido

Quando você atinge seu número máximo de 500 Java Virtual Machines, ainda pode aumentar sua grade. À medida que o número de Java Virtual Machines aumenta além de 500, a contagem de shards começa a diminuir abaixo de 10 para cada JVM, que está abaixo do número recomendado. Os shards começam a ficar maiores, o que pode causar problemas. Você deve repetir o processo de dimensionamento considerando novamente o crescimento futuro e reconfigurar a contagem da partição. Esta prática requer um reinício total da grade ou uma interrupção da sua grade.

Número de Servidores

Atenção: Não utilize paginação em um servidor sob nenhuma circunstância.

Uma única JVM utiliza mais memória do que o tamanho do heap. Por exemplo, 1 GB de heap para uma JVM na verdade utiliza 1.4 GB de memória real. Determine a RAM livre disponível no servidor. Divida a quantidade de RAM pela memória por JVM para obter o número máximo da Java Virtual Machines no servidor.

Dimensionando a CPU por Partição para Transações

Embora a maior funcionalidade do eXtreme Scale seja sua possibilidade de efetuar escala elástica, também é importante considerar o dimensionamento e o ajuste do número ideal de CPUs para efetuar scale up.

Os custos do processador incluem:

- Custo de serviços de operações de criação, recuperação, atualização e exclusão de clientes.
- Custo de replicação da outra Java Virtual Machines.
- Custo de invalidação.
- Custo da política de despejo.
- Custo da coleta de lixo.
- Custo da lógica de aplicativo.

Java Virtual Machines por servidor

Utilize dois servidores e inicie a contagem máxima de JVM por servidor. Utilize as contagens de partição calculadas da seção anterior. Em seguida, pré-carregue a Java Virtual Machines com dados suficientes para ajuste apenas nestes dois computadores. Utilize um servidor separado como um cliente. Execute uma simulação de transação realista junto a esta grade de dois servidores.

Para calcular a linha de base, tente saturar o uso do processador. Se não for possível saturar o processador, então, é provável que a rede esteja saturada. Se a rede estiver saturada, inclua mais cartões de rede e execute o round robin da Java Virtual Machines sobre os vários cartões de rede.

Execute os computadores em 60% de uso do processador, meça a taxa de transações de criação, recuperação, atualização e exclusão. Esta medida fornece o rendimento nos dois servidores. Este número duplica com quatro servidores, duplica novamente em 8 servidores e assim por diante. Esta escala assume que a capacidade da rede e a capacidade do cliente também tenham a capacidade de escalar.

Como resultado, o tempo de resposta do eXtreme Scale deve ser estável à medida que o número de servidores é escalado para cima. O rendimento da transação deve escalar linearmente à medida que os computadores são incluídos à grade.

Dimensionando CPUs para Transações Paralelas

As transações de partição única possuem escala de rendimento linear à medida que a grade cresce. As transações paralelas são diferentes das transações de partição única porque elas acessam um subconjunto de servidores, que poderiam ser todos os servidores.

Se uma transação acessar todos os servidores, o rendimento será limitado ao rendimento do cliente que inicia a transação ou ao servidor mais lento que está sendo acessado. Grades maiores propagam mais os dados e fornecem mais espaço do processador, memória, rede e assim por diante. Entretanto, o cliente deve aguardar que o servidor mais lento responda, e o cliente deve consumir os resultados da transação.

Quando uma transação acessa um subconjunto de servidores, M entre N servidores obtêm uma solicitação. O rendimento é então dividido N por M vezes mais rápido do que o rendimento do servidor mais lento. Por exemplo, se você tiver 20 servidores e uma transação que acessa 5 servidores, então, o rendimento é de 4 vezes o rendimento do servidor mais lento na grade.

Quando uma transação paralela é concluída, os resultados são enviados para o encadeamento do cliente que iniciou a transação. Este cliente deve então agregar os resultados únicos encadeados. Este tempo de agregação aumenta à medida que os número de servidores acessados pela transação aumentam. Entretanto, este tempo depende do aplicativo porque é possível que cada servidor retorne um resultado menor à medida que a grade aumenta.

Normalmente, as transações paralelas acessam todos os servidores na grade porque as partições são distribuídas de maneira uniforme pela grade. Neste caso, o rendimento é limitado ao primeiro caso.

Resumo

Com este dimensionamento, são obtidas três métricas como a seguir:

- Número de partições.
- Número de servidores que são necessários para a memória que é necessária.
- Número de servidores que são necessários para o rendimento necessário.

Se você precisar de 10 servidores para requisitos de memória mas estiver obtendo apenas 50% do rendimento necessário devido à saturação do processador, então precisa do dobro de servidores.

Para maior estabilidade, você deve executar seus servidores a um carregamento de processador de 60% e JVMheaps a um carregamento de heap de 60%. Os picos podem então conduzir o uso do processador a 80-90%, mas não execute seus servidores regularmente em níveis superiores a este.

Capítulo 4. Instalando e Implementando o WebSphere eXtreme Scale

O WebSphere eXtreme Scale é uma grade de dados de memória que pode ser usada para particionar, replicar e gerenciar dados e lógica de negócios dinamicamente através de vários servidores.

Antes de Iniciar

- Estabeleça como o WebSphere eXtreme Scale se enquadra na sua topologia atual. Consulte WebSphere eXtreme Scale no *Visão Geral do Produto* para obter mais informações.
- Verifique se o seu ambiente atende aos pré-requisitos para instalar o eXtreme Scale. Consulte "Requisitos de Hardware e Software" na página 7 para obter informações adicionais.

Por Que e Quando Desempenhar Esta Tarefa

Ambientes com Suporte

Não é necessário instalar e implementar o eXtreme Scale em um nível específico do sistema operacional. Cada instalação do Java Platform, Standard Edition e do Java Platform, Enterprise Edition requer níveis ou correções diferentes do sistema operacional.

É possível instalar e implementar o produto nos ambientes do Java EE e do Java SE. Também é possível incluir em pacote configurável o componente do cliente com os aplicativos Java EE diretamente sem integrar com o WebSphere Application Server. O eXtreme Scale suporta o Java Runtime Environment (JRE) Versão 1.4.2 e superior e o WebSphere Application Server Versão 6.0.2 e superior.

Instalação do eXtreme Scale independente

É possível instalar o eXtreme Scale independente em um ambiente que não contenha o WebSphere Application Server ou o WebSphere Application Server Network Deployment. Com a opção independente, defina um novo local de instalação no qual o servidor eXtreme Scale será instalado.

Integre o produto com o WebSphere Application Server ou o WebSphere Application Server Network Deployment

É possível instalar e integrar o eXtreme Scale com uma instalação existente do WebSphere Application Server ou do WebSphere Application Server Network Deployment.É possível instalar o cliente e o servidor do eXtreme Scale ou instalar apenas o cliente.

Criar e Alterar Perfis

Crie e aumente os perfis para usar os recursos do eXtreme Scale. Se você estiver executando o WebSphere Application Server Versão 6.1 ou Versão 7.0, é possível usar o plug-in Profile Management Tool ou o comando manageprofiles. Se você estiver executando o WebSphere Application Server Versão 6.0.2, é necessário usar o comando wasprofile para criar e aumentar os perfis.

Aplique manutenção

Use o IBM Update Installer Versão 7.0.0.4 ou superior para aplicar manutenção no seu ambiente.

Migrando para o WebSphere eXtreme Scale Versão 7.0

Para fazer upgrade da Versão 6.1.0.x para a Versão 7.0, mescle quaisquer arquivos de script de produto modificados com os novos arquivos de script do produto para manter as alterações.

Antes de Iniciar

Verifique se o sistemas atendem aos requisitos mínimos para as versões do produto que você planeja migrar e instalar. Consulte "Requisitos de Hardware e Software" na página 7 para obter mais informações.

Por Que e Quando Desempenhar Esta Tarefa

Mescle quaisquer arquivos de script de produto modificados com os novos arquivos de script do produto no diretório `/bin` para manter as alterações.

Dica: Se você não modificar os arquivos de script que estão instalados com o produto, não será necessário concluir as seguintes etapas de migração. Em vez disso, é possível fazer upgrade para a Versão 7.0 ao desinstalar a versão anterior e instalar a nova versão no mesmo diretório.

1. Pare todos os processos que usam o eXtreme Scale.
   - Consulte o "Parando Servidores eXtreme Scale Independentes" na página 228 para parar todos os processos em execução no ambiente do eXtreme Scale independente.
   - Consulte Utilitários da Linha de Comandos para parar todos os processos que estão em execução no ambiente do WebSphere Application Server ou do WebSphere Application Server Network Deployment.
2. Salve quaisquer scripts modificados a partir do diretório de instalação atual em um diretório temporário.
3. Desinstale o produto.
4. Instale o eXtreme Scale Versão 7.0. Consulte Capítulo 4, "Instalando e Implementando o WebSphere eXtreme Scale", na página 27 para obter informações adicionais.
5. Mescle suas alterações a partir dos arquivos no diretório temporário para os novos arquivos de script do produto no diretório `/bin`.
6. Inicie todos os processos do eXtreme Scale para começar a usar o produto. Consulte Capítulo 7, "Administrando o Ambiente", na página 215 para obter informações adicionais.

Instalando o WebSphere eXtreme Scale Independente

É possível instalar o WebSphere eXtreme Scale independente em um ambiente que não contenha o WebSphere Application Server ou o WebSphere Application Server Network Deployment.

Antes de Iniciar

- Verifique se o diretório de instalação de destino está vazio ou não existe.

Nota: Se uma versão anterior do eXtreme Scale ou do componente ObjectGrid existir no diretório no qual você especificar instalar a Versão 7.0, o produto não será instalado. Você pode escolher um diretório de instalação diferente ou cancelar a instalação. Em seguida, remova a instalação anterior e execute o assistente novamente.

- Para obter melhor desempenho e capacidade de manutenção, faça download e instale um IBM Developer Kit a partir do developerWorks.

  Restrição: Windows Se você estiver usando um hardware a partir de um fornecedor independente, selecione uma das seguintes opções para fazer download e instalar um Developer Kit:
  - Download do Sun JDK.
  - Download do JDK ou do JRE de outro fornecedor de software independente.

Por Que e Quando Desempenhar Esta Tarefa

Ao instalar o produto como independente, instale o cliente e o servidor do eXtreme Scale de modo independente. Com isso, os processos do servidor e do cliente acessam todos os recursos necessários localmente. Também é possível integrar o eXtreme Scale nos aplicativos Java Platform, Standard Edition existentes ao usar os scripts e arquivos Java archive (JAR).

A seguinte tabela lista os arquivos JAR incluídos na instalação.

Tabela 3. Arquivos de tempo de execução no diretório de instalação `/ObjectGrid/lib`

Nome do arquivo	Ambiente	Descrição
`cglib.jar`	Local, cliente e servidor	O arquivo `cglib.jar` é lido pela função de utilitário cglib quando estiver usando o modo de cópia copy-on-write e quando estiver usando o EntityManager para controlar as alterações de uma entidade. O arquivo é automaticamente incluído no tempo de execução do servidor quando usar os scripts fornecidos. Inclua esse arquivo no tempo de execução local ou do cliente.
`objectgrid.jar`	Local, cliente e servidor	O arquivo `objectgrid.jar` é usado pelo tempo de execução do servidor do Java Platform, Standard Edition Versão 1.4.2 e superior. O arquivo é automaticamente incluído no tempo de execução do servidor quando usar os scripts fornecidos.
`ogagent.jar`	Local, cliente e servidor	O arquivo `ogagent.jar` contém as classes do tempo de execução que são necessárias para executar o agente de instrumentação Java que é usado com a API do EntityManager.
`ogclient.jar`	Local e cliente	O arquivo `ogclient.jar` contém apenas o tempo de execução local e o tempo de execução do cliente. É possível utilizar este arquivo com o Java SE Versão 1.4.2 e posterior.
`wsogclient.jar`	Local e cliente	O arquivo `wsogclient.jar` é incluído ao instalar o produto em um ambiente que contém o WebSphere Application Server Versão 6.0.2 e superior. Este arquivo contém apenas os tempos de execução local e do cliente.
`wxsdynacache.jar`	Apenas servidor	O arquivo `wxsdynacache.jar` contém as classes necessárias a serem usadas com o provedor de cache dinâmico. O arquivo é automaticamente incluído no tempo de execução do servidor quando usar os scripts fornecidos. **Atenção:** O arquivo está no diretório `ObjectGrid/dynacache/lib`.

1. Use o assistente para concluir a instalação. Execute o seguinte script para iniciar o assistente:
   - Linux UNIX *dvd_root*`/install`
   - Windows *dvd_root*`\install.bat`
2. Siga os avisos no assistente e clique em **Concluir** para concluir a instalação.

   Nota: O painel de recursos opcionais lista os recursos a partir dos quais você pode escolher instalar. Porém, os recursos não poderão ser incluídos incrementalmente no ambiente do produto depois que o produto for instalado. Se você escolher não instalar um recurso com a instalação inicial do produto, deverá desinstalar e reinstalar o produto para incluir o recurso.

O que Fazer Depois

Configure os processos do aplicativo e do servidor. Consulte Capítulo 6, “Configurando WebSphere eXtreme Scale”, na página 59 para obter informações adicionais.

Usando o Object Request Broker com os Processos do WebSphere eXtreme Scale

É possível usar o WebSphere eXtreme Scale com aplicativos que usam o Object Request Broker (ORB) diretamente em ambientes que não contêm o WebSphere Application Server ou o WebSphere Application Server Network Deployment.

Antes de Iniciar

Se você usar o ORB dentro do mesmo processo do eXtreme Scale ao executar aplicativos ou outros componentes e estruturas que não estejam incluídos com o eXtreme Scale, poderá ser necessário concluir tarefas adicionais para garantir que o eXtreme Scale seja executado corretamente no seu ambiente.

Por Que e Quando Desempenhar Esta Tarefa

Inclua a propriedade ObjectGridInitializer no arquivo `orb.properties` para começar a usar o ORB no seu ambiente. Use o ORB para ativar a comunicação entre os processos do eXtreme Scale e outros processos que estão em seu ambiente. O arquivo `orb.properties` está no diretório `java/jre/lib`. Consulte o “Arquivo de Propriedades ORB” na página 197 para obter descrições das propriedades e configurações.

Digite a seguinte linha e salve as alterações:

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

Resultados

O eXtreme Scale inicializa corretamente o ORB e coexiste com outros aplicativos para os quais o ORB está ativado.

Para usar uma versão customizada do ORB com o eXtreme Scale, consulte “Configurando um Object Request Broker Customizado” na página 53.

Integrando o WebSphere eXtreme Scale com o WebSphere Application Server

É possível instalar o WebSphere eXtreme Scale em um ambiente no qual o WebSphere Application Server ou o WebSphere Application Server Network Deployment está instalado. É possível usar os recursos existentes do WebSphere Application Server ou do WebSphere Application Server Network Deployment para aprimorar os aplicativos ao aplicar a capacidade do eXtreme Scale.

Antes de Iniciar

- Instale WebSphere Application Server ou WebSphere Application Server Network Deployment. Consulte Instalando o Ambiente de Serviço do Aplicativo para obter mais informações.
- Com base na versão instalada, Versão 6.0.x, Versão 6.1 ou Versão 7.0, aplique o fix pack mais recente para o WebSphere Application Server ou o WebSphere

Application Server Network Deployment para atualizar o nível do produto. Consulte os Fix Packs mais Recentes para o WebSphere Application Server para obter mais informações.

- Verifique se o diretório de instalação de destino não contém uma instalação existente do eXtreme Scale.
- Pare todos os processos em execução no WebSphere Application Server ou no WebSphere Application Server Network Deployment. Consulte Utilitários de Linha de Comando para obter mais informações.

Por Que e Quando Desempenhar Esta Tarefa

Integre o eXtreme Scale com o WebSphere Application Server ou WebSphere Application Server Network Deployment para aplicar os recursos do eXtreme Scale nos aplicativos do Java Platform, Enterprise Edition. Os aplicativos do Java EE hospedam as grades do eXtreme Scale e acessam as grades usando uma conexão do cliente.

A seguinte tabela lista os arquivos Java archive (JAR) que são incluídos na instalação.

Tabela 4. Arquivos de tempo de execução no diretório de instalação /lib

Nome do arquivo	Ambiente	Descrição
cglib.jar	Local, cliente e servidor	O arquivo cglib.jar é lido pela função de utilitário cglib quando estiver usando o modo de cópia copy-on-write e quando estiver usando a API do EntityManager para controlar as alterações de uma entidade.
ogagent.jar	Local, cliente e servidor	O arquivo ogagent.jar contém as classes do tempo de execução que são necessárias para executar o agente de instrumentação Java que é usado com a API do EntityManager.
wsobjectgrid.jar	Local, cliente e servidor	O arquivo wsobjectgrid.jar contém os tempos de execução local, do cliente e do servidor do eXtreme Scale.
wsogclient.jar	Local e cliente	O arquivo wsogclient.jar é incluído ao instalar o produto em um ambiente que contém o WebSphere Application Server Versão 6.0.2 e superior. Este arquivo contém apenas os tempos de execução local e do cliente.
wxsdynacache.jar	Apenas servidor	O arquivo wxsdynacache.jar contém as classes necessárias a serem usadas com o provedor de cache dinâmico.

1. Use o assistente para concluir a instalação. Execute o seguinte script para iniciar o assistente:
   - Linux UNIX *dvd_root*/install
   - Windows *dvd_root*\install.bat
2. Siga os prompts no assistente.

   O painel de recursos opcionais lista os recursos a partir dos quais você pode escolher instalar. Porém, os recursos não poderão ser incluídos incrementalmente no ambiente do produto depois que o produto for instalado. Se você escolher não instalar um recurso com a instalação inicial do produto, deverá desinstalar e reinstalar o produto para incluir o recurso.

   O painel Aumento de Perfil lista os perfis existentes que podem ser selecionados para alterar com os recursos do eXtreme Scale. Porém, se você selecionar perfis existentes que já estão em uso, um painel de aviso será exibido. Para continuar com a instalação, pare os servidores que estão configurados nos perfis ou clique em **Voltar** para remover os perfis da sua seleção.

O que Fazer Depois

Se você estiver executando o WebSphere Application Server Versão 6.1 ou Versão 7.0, é possível usar o plug-in Profile Management Tool ou o comando

manageprofiles. Se você estiver executando o WebSphere Application Server Versão 6.0.2, é necessário usar o comando wasprofile para criar e alterar os perfis.

Implemente o aplicativo, inicie o serviço de catálogo e inicie os contêineres no seu ambiente do WebSphere Application Server. Consulte "Administrando o WebSphere eXtreme Scale com o WebSphere Application Server" na página 230 para obter informações adicionais.

Uso do Plug-in Installation Factory para Criação e Instalação de Pacotes Customizados

Utilize o plug-in do IBM Installation Factory para WebSphere eXtreme Scale para criar um pacote de instalação customizado (CIP) ou um pacote de instalação integrado (IIP). Um CIP contém um único pacote de instalação do produto e vários recursos opcionais. Um IIP combina um ou mais pacotes de instalação em um único fluxo de trabalho de instalação projetado.

Antes de Iniciar

Antes de poder criar e instalar pacotes customizados para o eXtreme Scale, primeiro é necessário fazer o download dos seguintes produtos:

- IBM Installation Factory para WebSphere Application Server
- Plug-in doIBM Installation Factory lug-in para o WebSphere eXtreme Scale

Por Que e Quando Desempenhar Esta Tarefa

Utilizando o Installation Factory, é possível criar um CIP ao combinar um único componente de produto com pacotes de manutenção, scripts de customização e outros arquivos. Ao criar um IIP, você agrega componentes individuais ou pacotes de instalação em um único pacote de instalação.

Arquivo de Definição da Construção

Um arquivo de definição de construção é um documento XML que especifica como construir e instalar um pacote de instalação customizado (CIP) ou um pacote de instalação integrado (IIP). O IBM Installation Factory para WebSphere eXtreme Scale lê os detalhes do pacote do arquivo de definição de construção para gerar um CIP ou um IIP.

Para poder criar um CIP ou um IIP, é necessário criar um arquivo de definição de construção para cada pacote customizado. O arquivo de definição de construção descreve quais componentes do produto ou pacotes de instalação devem ser instalados, o local do CIP ou do IIP, os pacotes de manutenção a serem incluídos, os scripts de instalação e outros arquivos escolhidos. Também é possível especificar no arquivo de definição de construção do IIP a ordem na qual o Installation Factory instalará cada pacote de instalação.

O assistente de Definição de Construção demonstra o processo de criação de um arquivo de definição de construção. Também é possível utilizar o assistente para modificar um arquivo de definição de construção existente. Cada painel do assistente de Definição de Construção solicita informações sobre um pacote customizado, como a identificação do pacote, o local de instalação da definição de construção e o local de instalação do pacote customizado. Todas essas informações são salvas no novo arquivo de definição de construção, ou modificadas e salvas em um arquivo de definição de construção existente. Para obter informações adicionais, consulte os painéis do Assistente de Definição de Construção do CIP e os painéis do Assistente de Definição de Construção do IIP.

Para criar apenas o arquivo de definição de construção, você pode utilizar a ferramenta da interface da linha de comandos para gerar o pacote customizado fora da GUI. Consulte “Instalação Silenciosa de um CIP ou IIP” na página 39 para obter informações adicionais.

Criando um Arquivo de Definição de Construção e Gerando um CIP

O plug-in do IBM Installation Factory para WebSphere eXtreme Scale gera um pacote de instalação customizado (CIP) de acordo com os detalhes especificados no arquivo de definição de construção. A definição de construção especifica o pacote do produto a ser instalado, o local do CIP, os pacotes de manutenção a serem incluídos na instalação, os arquivos de script de instalação e quaisquer arquivos adicionais a serem incluídos no CIP.

Por Que e Quando Desempenhar Esta Tarefa

É possível usar o assistente de definição de Construção para criar um arquivo de definições de construção e gerar um CIP.

1. Execute o seguinte script a partir do diretório *IF_HOME*/bin para iniciar o Installation Factory:
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`

   Clique no ícone **Nova Definição de Construção**.
2. Selecione o produto a ser incluído no arquivo de definições de construção e clique em **Concluir** para iniciar o assistente de definição de Construção.
3. Siga os prompts no assistente.

   No painel Scripts de Instalação e Desinstalação, clique em **Incluir Scripts...** para preencher a tabela com todos os scripts de instalação customizados. Digite o local dos arquivos de script e limpe a caixa de opção para continuar se uma mensagem de erro for exibida. A operação é interrompida por padrão. Clique em **OK** para retornar ao painel.

Resultados

Você criou e customizou o arquivo de definição de construção e gerou o CIP se tiver optado por trabalhar no modo conectado.

Se o assistente de definição de Construção não fornecer a opção de gerar o CIP a partir do arquivo de definições de construção, você ainda pode gerá-lo através da execução do script `ifcli.sh|bat` no diretório *IF_HOME*/bin.

O que Fazer Depois

Instale o CIP.

Instalando um CIP:

Simplifique o processo de instalação de produto instalando um pacote de instalação customizado (CIP - Customized Installation Package), que é uma única imagem de instalação de produto que pode incluir um ou mais pacotes de manutenção, scripts de configuração e outros arquivos.

Antes de Iniciar

Para poder instalar um CIP, é necessário criar um arquivo de definição de construção para especificar quais opções serão incluídas no CIP. Consulte o “Criando um Arquivo de Definição de Construção e Gerando um CIP” na página 33 para obter informações adicionais.

Por Que e Quando Desempenhar Esta Tarefa

Um CIP combina e instala um único componente do produto com pacotes de manutenção, scripts de customização e outros arquivos.

1. Pare todos os processos que estão sendo executados na estação de trabalho que você está preparando para instalação. Para parar o gerenciador de implementação, execute o seguinte script:
   - Linux UNIX *profile_root*/bin/stopManager.sh
   - Windows *profile_root*\bin\stopManager.bat

   Para parar os nós, execute o seguinte script:
   - Linux UNIX *profile_root*/bin/stopNode.sh
   - Windows *profile_root*\bin\stopNode.bat
2. Execute o seguinte script para iniciar a instalação:
   - Linux UNIX *CIP_home*/bin/install
   - Windows *CIP_home*\bin\install.bat
3. Siga os prompts no assistente para concluir a instalação.

   O painel de recursos opcionais lista os recursos a partir dos quais você pode escolher instalar. Porém, os recursos não poderão ser incluídos incrementalmente no ambiente do produto depois que o produto for instalado. Se você escolher não instalar um recurso com a instalação inicial do produto, deverá desinstalar e reinstalar o produto para incluir o recurso.

   O painel Aumento de Perfil lista os perfis existentes que podem ser selecionados para alterar com os recursos do eXtreme Scale. Porém, se você selecionar perfis existentes que já estão em uso, um painel de aviso será exibido. Para continuar com a instalação, pare os servidores que estão configurados nos perfis ou clique em **Voltar** para remover os perfis da sua seleção.

Resultados

Você instalou com êxito o CIP.

O que Fazer Depois

Se você estiver executando o WebSphere Application Server Versão 6.1 ou Versão 7.0, é possível usar o plug-in da Profile Management Tool ou o comando manageprofiles para criar e alterar perfis. Se você estiver executando o WebSphere Application Server Versão 6.0.2, é necessário usar o comando wasprofile para criar e aumentar os perfis. Consulte “Criando e Alterando Perfis para o WebSphere eXtreme Scale” na página 40 para obter informações adicionais.

Se você alterar perfis do eXtreme Scale durante o processo de instalação, poderá implementar os aplicativos, iniciar um serviço de catálogo e iniciar os contêineres no seu ambiente do WebSphere Application Server. Consulte “Administrando o

WebSphere eXtreme Scale com o WebSphere Application Server" na página 230 para obter mais informações.

Instalando um CIP para Aplicar Manutenção a uma Instalação do Produto Existente:

Você pode aplicar pacotes de manutenção a uma instalação existente do produto instalando um CIP (Pacote de Instalação Customizada). O processo de aplicar a manutenção a uma instalação existente com uma CIP é normalmente referida como *instalação slip*.

Antes de Iniciar

Crie um arquivo de definições de construção para especificar quais opções incluir no CIP. Consulte o "Criando um Arquivo de Definição de Construção e Gerando um CIP" na página 33 para obter informações adicionais.

Por Que e Quando Desempenhar Esta Tarefa

Ao aplicar a manutenção com um CIP que contém um pacote de atualizações, um fix pack ou ambos, todos os relatórios de análise do programa autorizado instalados anteriormente (APAR) são desinstalados pelo assistente. Se o CIP estiver no mesmo nível do produto, as APARs anteriormente instaladas permanecem somente se elas forem empacotadas no CIP. Para aplicar com êxito a manutenção em uma instalação existente, é necessário incluir os recursos instalados no CIP.

1. Pare todos os processos que estão sendo executados na estação de trabalho que você está preparando para instalação. Para parar o gerenciador de implementação, execute o seguinte comando:
   - Linux UNIX *profile_root*/bin/stopManager.sh
   - Windows *profile_root*\bin\stopManager.bat

   Para parar os nós, execute o seguinte script:
   - Linux UNIX *profile_root*\bin\stopNode.sh
   - Windows *profile_root*\bin\stopNode.bat
2. Execute o seguinte script para iniciar a instalação:
   - Linux UNIX *CIP_home*/bin/install
   - Windows *CIP_home*\bin\install.bat
3. Siga os prompts no assistente para concluir a instalação.

   O resumo da visualização da instalação relaciona a versão do produto resultante e todos os recursos aplicáveis e correções temporárias. Em seguida, o assistente aplica com êxito a manutenção e atualiza os recursos do produto.

Resultados

Os arquivos binários do produto são copiados para o diretório *was_home*/properties/version/nif/backup. É possível usar o IBM Update Installer para desinstalar a atualização e restaurar sua estação de trabalho. Consulte o "Desinstalando Atualizações de CIP de uma Instalação Existente do Produto" para obter informações adicionais.

Desinstalando Atualizações de CIP de uma Instalação Existente do Produto:

É possível remover atualizações de CIP de uma instalação de produto existente sem remover o produto todo. Use o IBM Update Installer Versão 7.0.0.4 para desinstalar quaisquer atualizações do CIP. Esta tarefa também é chamada de uma *desinstalação slip*.

Antes de Iniciar

Você precisa ter pelo menos uma cópia existente do produto instalada no sistema.

1. Faça Download da Versão 7.0.0.4 do Update Installer a partir do seguinte site FTP:

   ftp://ftp.software.ibm.com/software/websphere/cw/process_server/FEP/UPDI/7004
2. Instale o Update Installer. Consulte Instalando o Update Installer para WebSphere Software no Centro de Informações do WebSphere Application Server para obter informações adicionais.
3. Desinstale todos os fix packs, pacotes de atualizações ou correções temporárias que você incluiu em seu ambiente após a instalação do CIP.
4. Desinstale todas as correções temporárias que tiver incluído na instalação slip. Este processo é igual a desinstalar um fix pack simples ou pacote de atualizações. Porém, a manutenção que foi incluída no CIP agora está incluída em uma única operação.
5. Desinstale o CIP utilizando o Update Installer. Os níveis de manutenção retornam ao estado pré-atualização e o CIP é indicado pelo identificador CIP pré-anexado em seu nome do arquivo. O exemplo a seguir mostra como um CIP pode ser exibido de forma diferente de outros pacotes de manutenção regulares no painel de seleção do pacote de manutenção:

   CIP

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

Resultados

Você removeu com êxito as atualizações de CIP de uma instalação existente do produto.

Criando um Arquivo de Definição de Construção e Gerando um IIP

O plug-in IBM Installation Factory para o WebSphere eXtreme Scale gera um IIP baseado nas propriedades que o arquivo de definições de construção fornece, como quais pacotes de instalação devem ser incluídos no IIP, a ordem na qual o Installation Factory instala cada pacote e o local do IIP.

Por Que e Quando Desempenhar Esta Tarefa

É possível usar o assistente de definição de Construção para criar um arquivo de definições de construção e gerar um IIP.

1. Execute o seguinte script a partir do diretório *IF_HOME*/bin para iniciar o Installation Factory:
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`
2. Clique no ícone **Criar Novo Pacote de Instalação Integrado** para iniciar o assistente de Definição de Construção.
3. Siga os prompts no assistente.

a. No painel Construir o IIP, selecione um pacote de instalação suportado na lista, e clique em **Incluir Instalador** para incluir o pacote de instalação no IIP. Um painel com o nome do pacote, identificador do pacote e as propriedades do pacote é exibido. Para visualizar informações específicas sobre o pacote selecionado, clique em **Visualizar Informações do Pacote de Instalação**. Clique em **Modificar** para digitar o caminho de diretório para o pacote de instalação de cada sistema operacional. Se estiver incluindo um pacote de instalação para WebSphere Extended Deployment, selecione a caixa de opção, que fornece a opção para usar o mesmo pacote para todos os sistemas operacionais suportados. Clique em **OK** e retorne ao painel Construir o IIP. Uma chamada é criada por padrão.
   - Para modificar o caminho do diretório para um pacote de instalação, selecione o pacote a partir dos Pacotes de Instalação usados na lista do IIP e clique em **Modificar**.
   - Para modificar uma chamada, selecione a chamada e clique em **Modificar**. Especifique o local de instalação padrão da chamada em cada sistema operacional. Especifique o local para o arquivo de resposta se selecionar uma instalação silenciosa como o modo de instalação padrão.
   - Clique em **Incluir Chamada** para incluir uma contribuição de chamada no pacote de instalação. É exibido um painel a partir do qual você pode especificar propriedades para a chamada.
   - Clique em **Remover** para remover os pacotes de instalação ou as chamadas.

4. Revise o resumo das seleções, selecione a opção **Salvar arquivo de definição de construção e gerar pacote de instalação integrado** e clique em **Concluir**.

   Alternativamente, você pode optar por salvar o arquivo de definição de construção sem gerar o IIP. Com essa opção, você realmente gera o IIP fora do assistente executando o script `ifcli.bat | ifcli.sh` a partir do diretório *IF_home*/bin/.

Resultados

Você criou e customizou o arquivo de definições de construção para um IIP.

O que Fazer Depois

Instale o IIP.

Instalando um IIP:

Utilize o plug-in do IBM Installation Factory para WebSphere eXtreme Scale para instalar um pacote de instalação integrado (IIP). Um IIP combina um ou mais pacotes de instalação em um único fluxo de trabalho que você projeta.

Antes de Iniciar

Para poder instalar um CIP, é necessário criar um arquivo de definição de construção para especificar quais opções serão incluídas no CIP. Consulte o “Criando um Arquivo de Definição de Construção e Gerando um IIP” na página 36 para obter informações adicionais.

Por Que e Quando Desempenhar Esta Tarefa

Um IIP pode incluir um ou mais pacotes de instalação geralmente disponíveis, um ou mais CIPs e outros arquivos e diretórios opcionais. Ao instalar um IIP, agregue

vários pacotes de instalação ou *contribuições*, em um único pacote e instale as contribuições em uma ordem específica para concluir uma instalação de ponta a ponta.

1. Execute o seguinte script para iniciar o assistente:
   - Linux UNIX `IIP_home/bin/install`
   - Windows `IIP_home\bin\install.bat`
2. Clique em **Sobre** no painel Boas-vindas para visualizar os detalhes do IIP, como o identificador do pacote, os sistemas operacionais suportados e os pacotes de instalação incluídos.

   Optional: Para modificar as opções de instalação de cada pacote, clique em **Modificar**.

   Optional: Dois botões **Visualizar Log** são exibidos no painel do assistente. Para visualizar o log de cada pacote, clique no botão **Visualizar Log** exibido ao lado da tabela que lista os pacotes de instalação. Para visualizar os detalhes gerais do log do IIP, clique no botão **Visualizar Log** exibido ao lado das informações de status.
3. Selecione os pacotes de instalação a serem executados e clique em **Instalar**. É exibida uma lista de todas as contribuições na ordem de chamada que o IIP contém. Para designar que chamadas de contribuição não devem ser executadas durante a instalação, desmarque a caixa de opção localizada próxima ao campo **Nome da instalação**.

Resultados

Você instalou com êxito um IIP.

Modificando um Arquivo de Definição de Construção Existente de um IIP:

É possível editar ou incluir nas propriedades de um IIP para customizar ainda mais a instalação.

Por Que e Quando Desempenhar Esta Tarefa

Para alterar as propriedades de um IIP, modifique o arquivo de definição de construção existente.

1. Execute o seguinte script a partir do diretório *IF_HOME*/bin para iniciar o Installation Factory:
   - UNIX Linux `ifgui.sh`
   - Windows `ifgui.bat`
2. Clique no ícone **Abrir Definição de Construção** e selecione o arquivo de definições de construção que deseja modificar.
3. Selecione as propriedades específicas do IIP que deseja modificar. A lista a seguir contém as possíveis modificações que podem ser feitas:
   - Alterar a seleção de modo atual. No modo conectado, crie a definição de construção que utilizará e, opcionalmente, gere o IIP, a partir da estação de trabalho atual. No modo desconectado, crie o arquivo de definição de construção a ser utilizado em outra estação de trabalho.
   - Incluir ou remover os sistemas operacionais existentes suportados pelo IIP.
   - Editar o identificador e a versão existentes do IIP.
   - Editar o local de destino do arquivo de definição de construção.

- Editar o local de destino do IIP.
- Mudar se um assistente de instalação deve ser exibido para o IIP. O assistente fornece informações sobre o IIP e as opções de instalação quando o IIP é executado.
- Incluir, remover e editar os pacotes de instalação contidos no IIP.

  Importante: Se você tiver incluído um sistema operacional suportado e não tiver atualizado as propriedades do pacote de instalação no IIP, será exibida uma mensagem de aviso informando que as contribuições selecionadas não contêm os pacotes de instalação identificados de todos os sistemas operacionais suportados pelo IIP. Clique em **Sim** para continuar ou em **Não** para editar o pacote de instalação.

4. Revise o resumo das seleções, selecione **Salvar arquivo de definição de construção e gerar pacote de instalação integrado** e clique em **Concluir**.

Instalação Silenciosa de um CIP ou IIP

É possível instalar silenciosamente um pacote de instalação customizado (CIP) ou um pacote de instalação integrado (IIP) para o produto usando um arquivo de resposta completo, que é configurado especificamente para atender suas necessidades, ou parâmetros passados para a linha de comandos.

Antes de Iniciar

Crie o arquivo de definições de construção para o CIP ou IIP. Consulte o "Criando um Arquivo de Definição de Construção e Gerando um CIP" na página 33 para obter informações adicionais.

Por Que e Quando Desempenhar Esta Tarefa

Uma instalação silenciosa utiliza o mesmo programa que a versão de interface gráfica com o usuário (GUI) utiliza. Entretanto, no lugar de exibir uma interface de assistente, a instalação silenciosa lê todas as suas respostas de um arquivo que você customiza ou dos parâmetros que você transmite para a linha de comandos. Se estiver instalando silenciosamente um IIP, é possível chamar uma contribuição com uma combinação de opções especificadas diretamente na linha de comandos, bem como opções que você especifica em um arquivo de resposta. No entanto, as opções de contribuição transmitidas à linha de comandos farão com que o instalador do IIP ignore todas as opções especificadas em um arquivo de resposta específico da contribuição. Consulte as opções de instalação do IIP em detalhes para obter informações adicionais.

Nota: É necessário especificar o nome completo do arquivo de resposta. Especificar o caminho relativo faz com que a instalação falhe sem nenhuma indicação de que ocorreu um erro.

1. Opcional: Se você optar por instalar o CIP ou IIP usando um arquivo de resposta, primeiro customize o arquivo.
   a. Copie o arquivo de resposta `wxssetup.response.txt` do DVD do produto para a unidade de disco.
   b. Abra e edite o arquivo de resposta no editor de texto de sua escolha. O arquivo inclui comentários para ajudá-lo no processo de configuração e deve incluir estes parâmetros:
      - O contrato de licença
      - O local da instalação do produto

Dica: O instalador usa o local que você seleciona para sua instalação para determinar onde sua instância do WebSphere Application Server está instalada. Se você instalar em um nó com múltiplas instâncias do WebSphere Application Server, defina claramente seu local.

c. Execute o script a seguir para iniciar seu arquivo de resposta customizado.

- Linux UNIX `install -options /absolute_path/response_file.txt -silent`
- Windows `install.bat -options C:\drive_path\response_file.txt -silent`

2. Opcional: Se você optar por instalar o CIP ou IIP passando determinados parâmetros para a linha de comandos, execute o script a seguir para iniciar a instalação:

- Linux UNIX `install -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=install_location`
- Windows `install.bat -silent -OPT silentInstallLicenseAcceptance=true -OPT installLocation=install_location`

em que *install_location* é o local de sua instalação do WebSphere Application Server existente.

3. Revise os registros de erros resultantes ou uma falha na instalação.

Resultados

Você instalou silenciosamente o CIP ou IIP.

O que Fazer Depois

Se você estiver executando o WebSphere Application Server Versão 6.1 ou Versão 7.0, é possível usar o plug-in ferramenta Gerenciamento de Perfil ou o comando manageprofiles para criar e alterar perfis. Se você estiver executando o WebSphere Application Server Versão 6.0.2, é necessário usar o comando wasprofile para criar e alterar os perfis.

Se você alterar perfis do eXtreme Scale durante o processo de instalação, poderá implementar os aplicativos, iniciar um serviço de catálogo e iniciar os contêineres no seu ambiente do WebSphere Application Server. Consulte "Administrando o WebSphere eXtreme Scale com o WebSphere Application Server" na página 230 para obter mais informações.

Criando e Alterando Perfis para o WebSphere eXtreme Scale

Depois de instalar o produto, crie tipos exclusivos de perfis e aumente os perfis existentes para o WebSphere eXtreme Scale.

Antes de Iniciar

Instale o WebSphere eXtreme Scale. Consulte "Integrando o WebSphere eXtreme Scale com o WebSphere Application Server" na página 30 para obter mais informações.

Por Que e Quando Desempenhar Esta Tarefa

Executando com o WebSphere Application Server Versão 6.0.2

Se o seu ambiente contiver o WebSphere Application Server Versão 6.0.2, use o comando wasprofile para criar e alterar os perfis para o WebSphere eXtreme Scale como mostra o seguinte exemplo:

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

Consulte o Comando wasprofile no Centro de Informações do WebSphere Application Server para obter mais informações.

Executando com o WebSphere Application Server Versão 6.1 ou 7.0

Se o seu ambiente contiver o WebSphere Application Server Versão 6.1 ou Versão 7.0, poderá usar o plug-in da Profile Management Tool ou o comando manageprofiles para criar e alterar perfis.

O que Fazer Depois

Dependendo de qual tarefa você escolher concluir, ative o Console de Primeiras Etapas para obter assistência sobre a configuração e o teste do ambiente do produto. Como alternativa, repita qualquer uma das etapas anteriores para criar ou aumentar perfis adicionais.

Usando a Interface Gráfica com o Usuário para Criar Perfis

Use a interface gráfica com o usuário (GUI), que é fornecida pelo plug-in Profile Management Tool, para criar perfis para o WebSphere eXtreme Scale. Um perfil é um conjunto de arquivos que definem o ambiente de tempo de execução.

Antes de Iniciar

Nota: Se você estiver executando o WebSphere Application Server Versão 6.0.2 ou o WebSphere Application Server Network Deployment Versão 6.0.2, é necessário usar o comando wasprofile para criar ou aumentar um perfil para o WebSphere eXtreme Scale, como mostra o seguinte exemplo:

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

Consulte o Comando wasprofile no Centro de Informações do WebSphere Application Server para obter mais informações.

Por Que e Quando Desempenhar Esta Tarefa

Para usar os recursos do produto, o plug-in ferramenta de Gerenciamento de Perfil ativa a GUI para ajudá-lo a configurar perfis, como um perfil do WebSphere Application Server, um perfil do gerenciador de implementação, um perfil de célula e um perfil customizado.

Use a GUI do Profile Management Tool para criar perfis. Escolha uma das seguintes opções para iniciar o assistente:

- Selecione **Profile Management Tool** no console do First Steps.
- Acesse o Profile Management Tool a partir do menu **Iniciar**.
- Execute o script ./pmt.sh|bat no diretório *install_root*/bin/ProfileManagement.

A página Seleção de Ação é exibida apenas se, pelo menos, um perfil e os modelos de aumento existirem.

O que Fazer Depois

É possível criar perfis adicionais ou alterar perfis existentes. Para reiniciar o Profile Management Tool, execute o comando ./pmt.sh|bat no diretório *install_root*/bin/ProfileManagement ou selecione **Profile Management Tool** no console do First Steps.

Inicie um serviço de catálogo, inicie os contêineres e configure as portas TCP no seu ambiente do WebSphere Application Server. Consulte "Administrando o WebSphere eXtreme Scale com o WebSphere Application Server" na página 230 para obter mais informações.

Usando a Interface Gráfica com o Usuário para Alterar Perfis

Depois de instalar o produto, poderá alterar um perfil existente para torná-lo compatível com o WebSphere eXtreme Scale.

Antes de Iniciar

Nota: Se você estiver executando o WebSphere Application Server Versão 6.0.2 ou o WebSphere Application Server Network Deployment Versão 6.0.2, será necessário usar o comando wasprofile para criar ou alterar um perfil para o WebSphere eXtreme Scale, como mostra o seguinte exemplo:

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

Consulte o Comando wasprofile no Centro de Informações do WebSphere Application Server para obter mais informações.

Por Que e Quando Desempenhar Esta Tarefa

Ao alterar um perfil existente, poderá alterar o perfil ao aplicar um modelo de aumento específico do produto. Por exemplo, os servidores do WebSphere eXtreme Scale não são iniciados automaticamente a menos que o perfil do servidor seja alterado com o modelo xs_augment.

- Altere o perfil com o modelo xs_augment se você instalou o cliente ou o cliente e o servidor do eXtreme Scale.
- Altere o perfil com o modelo pf_augment se você instalou apenas o recurso de particionamento.
- Aplique ambos os modelos se o seu ambiente contiver o cliente e o recurso de particionamento do eXtreme Scale.

Use a GUI do Profile Management Tool para alterar os perfis para o eXtreme Scale. Escolha uma das seguintes opções para iniciar o assistente:

- Selecione **Profile Management Tool** no console do First Steps.
- Acesse o Profile Management Tool a partir do menu **Iniciar**.
- Execute o script ./pmt.sh|bat no diretório *install_root*/bin/ProfileManagement.

O que Fazer Depois

Você pode alterar perfis adicionais. Para reiniciar o Profile Management Tool, execute o comando ./pmt.sh|bat no diretório *install_root*/bin/ProfileManagement ou selecione **Profile Management Tool** no console do First Steps.

Inicie um serviço de catálogo, inicie os contêineres e configure as portas TCP no seu ambiente do WebSphere Application Server. Consulte “Administrando o WebSphere eXtreme Scale com o WebSphere Application Server” na página 230 para obter mais informações.

Comando manageprofiles

É possível usar o utilitário manageprofiles para criar perfis com o modelo WebSphere eXtreme Scale e alterar ou retornar o estado inalterado dos perfis do servidor de aplicativos existentes com os modelos de alteração do eXtreme Scale. Para usar os recursos do produto, seu ambiente deve conter pelo menos um perfil alterado para o produto.

- Antes de poder criar e alterar os perfis, é necessário instalar o eXtreme Scale . Consulte “Integrando o WebSphere eXtreme Scale com o WebSphere Application Server” na página 30 para obter informações adicionais.
- Se o seu ambiente contiver o WebSphere Application Server Versão 6.0.2, é necessário usar o comando wasprofile para criar e alterar os perfis para o eXtreme Scale como mostra o seguinte exemplo:

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

  Consulte o Comando wasprofile no Centro de Informações do WebSphere Application Server para obter mais informações.

Propósito

O comando manageprofiles cria o ambiente de execução para um processo do produto em um conjunto de arquivos chamado perfil. O perfil define o ambiente de tempo de execução. É possível executar as seguintes ações com o comando manageprofiles:

- Criar e alterar um perfil do gerenciador de implementação
- Criar e alterar um perfil customizado
- Criar e alterar um perfil do servidor de aplicativos independente
- Criar e alterar um perfil de célula
- Retornar o estado inalterado de qualquer tipo de perfil

Ao alterar um perfil existente, poderá mudar o perfil ao aplicar um modelo de alteração específico do produto.

- Altere o perfil com o modelo xs_augment se você instalou o cliente ou o cliente e o servidor do eXtreme Scale.
- Altere o perfil com o modelo pf_augment se você instalou apenas o recurso de particionamento.
- Aplique os dois modelos se o seu ambiente contiver o cliente e o recurso de particionamento do eXtreme Scale.

Local

O arquivo de comando está no diretório *install_root*/bin.

Uso

Para obter ajuda detalhada, use o parâmetro **-help**:

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

Nas seguintes seções, cada tarefa que pode ser executada usando o comando manageprofiles junto com uma lista de parâmetros necessários é descrita. Para

obter detalhes sobre os parâmetros opcionais para especificar cada tarefa, consulte o comando manageprofiles no Centro de Informações do WebSphere Application Server.

Criar um Perfil do Gerenciador de Implementação

É possível usar o comandomanageprofiles para criar um perfil do gerenciador de implementação. O gerenciador de implementação administra os servidores de aplicativos que são federados na célula.

Parâmetros

-create
Cria um perfil. (Requerido)

-templatePath *template_path*
Especifica o caminho de arquivo para o modelo. (Obrigatório)

Utilize o seguinte formato:

```
-templatePath install_root/profileTemplates/template_type/dmgr
```

em que *template_type* é `xs_augment` ou `pf_augment`.

Exemplo

- Usando o modelo xs_augment:

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

- Usando o modelo pf_augment:

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

Criar um perfil customizado

É possível usar o comandomanageprofiles para Criar um perfil customizado. Um perfil customizado é um nó vazio customizado por meio do gerenciador de implementação para incluir servidores de aplicativos, clusters ou outros processos Java.

Parâmetros

-create
Cria um perfil. (Requerido)

-templatePath *template_path*
Especifica o caminho de arquivo para o modelo. (Obrigatório)

Utilize o seguinte formato:

```
-templatePath install_root/profileTemplates/template_type/managed
```

em que *template_type* é `xs_augment` ou `pf_augment`.

Exemplo

- Usando o modelo xs_augment:

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

- Usando o modelo pf_augment:

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

Criar um Perfil de Servidor de Aplicativos Independente

É possível usar o comandomanageprofiles para Criar um perfil do servidor de aplicativos independente.

Parameters

-create
Cria um perfil. (Requerido)

-templatePath *template_path*
Especifica o caminho de arquivo para o modelo. (Obrigatório)

Utilize o seguinte formato:

```
-templatePath install_root/profileTemplates/template_type/default
```

em que *template_type* é `xs_augment` ou `pf_augment`.

Exemplo

- Usando o modelo xs_augment:

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

- Usando o modelo pf_augment:

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

Criar o Perfil de uma Célula

É possível usar o comandomanageprofiles para Criar um perfil de célula que consiste de um gerenciador de implementação e de um servidor de aplicativos.

Parameters

Especifique os seguintes parâmetros no modelo do gerenciador de implementação:

-create
Cria um perfil. (Requerido)

-templatePath *template_path*
Especifica o caminho de arquivo para o modelo. (Obrigatório)

Utilize o seguinte formato:

```
-templatePath install_root/profileTemplates/template_type/cell/dmgr
```

em que *template_type* é `xs_augment` ou `pf_augment`.

Especifique os seguintes parâmetros com o modelo do servidor de aplicativos:

-create
Cria um perfil. (Requerido)

-templatePath *template_path*
Especifica o caminho de arquivo para o modelo. (Obrigatório)

Utilize o seguinte formato:

```
-templatePath install_root/profileTemplates/template_type/cell/default
```

em que *template_type* é `xs_augment` ou `pf_augment`.

Exemplo

- Usando o modelo xs_augment:

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

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

- Usando o modelo pf_augment:

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

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

Alterar um Perfil do Gerenciador de Implementação

É possível usar o comandomanageprofiles para Alterar um perfil do gerenciador de implementação.

Parameters

-augment
Altera o perfil existente. (Requerido)

-profileName
Especifica o nome do perfil. (Requerido)

-templatePath *template_path*
Especifica o caminho para os arquivos de modelo que estão localizados no diretório raiz da instalação. (Obrigatório)

Utilize o seguinte formato:

```
-templatePath install_root/profileTemplates/template_type/dmgr
```

em que *template_type* é `xs_augment` ou `pf_augment`.

Exemplo

- Usando o modelo xs_augment:

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

- Usando o modelo pf_augment:

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

Alterar um Perfil Customizado

É possível usar o comandomanageprofiles para Alterar um perfil customizado.

Parameters

-augment
Altera o perfil existente. (Requerido)

-profileName
Especifica o nome do perfil. (Requerido)

-templatePath *template_path*
Especifica o caminho para os arquivos de modelo que estão localizados no diretório raiz da instalação. (Obrigatório)

Utilize o seguinte formato:

```
-templatePath install_root/profileTemplates/template_type/managed
```

em que *template_type* é `xs_augment` ou `pf_augment`.

Exemplo

- Usando o modelo xs_augment:

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

- Usando o modelo pf_augment:

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

Alterar um Perfil de Servidor de Aplicativos Independente

É possível usar o comandomanageprofiles para Alterar um perfil do servidor de aplicativos independente.

Parameters

-augment
Altera o perfil existente. (Requerido)

-profileName
Especifica o nome do perfil. (Requerido)

-templatePath *template_path*
Especifica o caminho para os arquivos de modelo que estão localizados no diretório raiz da instalação. (Obrigatório)

Utilize o seguinte formato:

```
-templatePath install_root/profileTemplates/template_type/default
```

em que *template_type* é `xs_augment` ou `pf_augment`.

Exemplo

- Usando o modelo xs_augment:

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

- Usando o modelo pf_augment:

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

Alterar o Perfil de uma Célula

É possível usar o comandomanageprofiles para Alterar um perfil de célula.

Parameters

Especifique os seguintes parâmetros para o perfil do gerenciador de implementação:

-augment
Altera o perfil existente. (Requerido)

-profileName
Especifica o nome do perfil. (Requerido)

-templatePath *template_path*
Especifica o caminho para os arquivos de modelo que estão localizados no diretório raiz da instalação. (Obrigatório)

Utilize o seguinte formato:

```
-templatePath install_root/profileTemplates/template_type/cell/dmgr
```

em que *template_type* é `xs_augment` ou `pf_augment`.

Especifique os seguintes parâmetros com o perfil do servidor de aplicativos:

-augment
Altera o perfil existente. (Requerido)

-profileName
Especifica o nome do perfil. (Requerido)

-templatePath *template_path*
Especifica o caminho para os arquivos de modelo que estão localizados no diretório raiz da instalação. (Obrigatório)

Utilize o seguinte formato:

```
-templatePath install_root/profileTemplates/template_type/cell/default
```

em que *template_type* é `xs_augment` ou `pf_augment`.

Exemplo

- Usando o modelo xs_augment:

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

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

- Usando o modelo pf_augment:

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

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

Retornar o Estado Inalterado de um Perfil

Para reverter o aumento de um perfil, especifique o parâmetro **-ignoreStack** com o parâmetro **-templatePath** além de especificar os parâmetros requiridos **-unaugment** e **-profileName**.

Parameters

-unaugment
Retorna o estado inalterado de um perfil alterado anteriormente. (Requerido)

-profileName
Especifica o nome do perfil. O parâmetro é emitido por padrão se nenhum valor for especificado. (Requerido)

-templatePath *template_path*
Especifica o caminho para os arquivos de modelo que estão localizados no diretório raiz da instalação. (Opcional)

Utilize o seguinte formato:

```
-templatePath install_root/profileTemplates/template_type/profile_type
```

em que *template_type* é `xs_augment` ou `pf_augment` e *profile_type* é um dos quatro tipos de perfil:

- `dmgr`: perfil do gerenciador de implementação
- `managed`: perfil customizado
- `default`: perfil do servidor de aplicativos independente
- `cell`: perfil de célula

-ignoreStack
Usado com o parâmetro **-templatePath** para alterar um perfil específico que foi alterado. (Opcional)

Exemplo

- Usando o modelo xs_augment:

```
./manageprofile.sh|bat -unaugment -profileName profile01 -ignoreStack
-templatePath install_root/profileTemplates/xs_augment/profile_type
```

- Usando o modelo pf_augment:

```
./manageprofile.sh|bat -unaugment -profileName profile01 -ignoreStack
-templatePath install_root/profileTemplates/pf_augment/profile_type
```

Perfis Não-root

Dá permissões a um usuário não-root para arquivos e diretórios para que o usuário não-root possa criar um perfil para o produto. O usuário não-root também pode alterar um perfil que foi criado por um usuário root, um usuário não-root diferente ou pelo mesmo usuário não-root.

Em geral, os usuários não-root são limitados na capacidade de criar e utilizar perfis em seus ambientes. Dentro do plug-in da ferramenta de Gerenciamento de Perfil, os nome exclusivos e os valores da porta são desativados para usuários não-raiz. O usuário não-root deve alterar os valores de campo padrão na ferramenta Gerenciamento de Perfil para o nome do perfil, o nome do nó, o nome da célula e as designações de porta. Considere designar aos usuários não-root um intervalo de valores para cada um dos campos. É possível designar responsabilidade para os usuários não-root para aderirem a seus intervalos de valor apropriados e para manterem a integridade de suas próprias definições.

O termo *instalador* refere-se a um usuário root ou não-root. Como um instalador, você pode conceder permissões de usuários não-root para criar e estabelecer seus próprios ambientes de produto. Por exemplo, um usuário não-root pode criar um ambiente de produto para testar a implementação do aplicativo com um perfil que ele possui. Tarefas específicas que você pode concluir para permitir a criação de perfil não-root incluem os seguintes itens:

- Criar um perfil e designar a propriedade do diretório de perfil para um usuário não-raiz, para que o usuário não-raiz possa iniciar o WebSphere Application Server para um perfil específico.
- Conceder permissão de gravação dos arquivos e diretórios apropriados para um usuário não-root, permitindo que o usuário não-root crie o perfil. Com essa tarefa, você pode criar um grupo para usuários autorizados a criar perfis ou pode fornecer aos usuários individuais a capacidade de criar perfis.
- Instalar os pacotes de manutenção para o produto que inclui o serviço necessário para os perfis existentes de propriedade de um usuário não-root. Como o instalador, você é o proprietário de qualquer novo arquivo criado pelo pacote de manutenção.

Para obter detalhes adicionais, leia as informações detalhadas sobre a criação de perfis para usuários não-root, o que inclui as etapas para concluir os exemplos de tarefas precedentes, no Centro de Informações do WebSphere Application Server Network Deployment.

Como um instalador, você também pode conceder permissões para um usuário não-root alterar perfis. Por exemplo, um usuário não-root pode alterar um perfil criado por um instalador ou alterar um perfil que ele mesmo cria. Siga o processo de aumento do usuário não-raiz do WebSphere Application Server Network Deployment para executar essas tarefas.

Porém, quando um usuário não-root altera um perfil criado pelo instalador, os arquivos a seguir não precisam ser criados pelo usuário não-root antes da alteração, pois os arquivos estabelecidos durante o processo de criação do perfil:

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

Quando um usuário não-raiz altera perfil criado por ele, o usuário não-raiz deve modificar as permissões para os documentos que estão localizados dentro dos modelos de perfil do eXtreme Scale.

Instalando o WebSphere eXtreme Scale Silenciosamente

Utilize um arquivo de resposta completo, que pode ser configurado especificamente às suas necessidades, ou passar os parâmetros para a linha de comandos para instalar o WebSphere eXtreme Scale silenciosamente.

Antes de Iniciar

Pare todos os processos em execução no ambiente do WebSphere Application Server ou do WebSphere Application Server Network Deployment. Consulte Utilitários da Linha de Comandos para obter mais informações sobre os comandos stopManager, stopNode e stopServer.

Por Que e Quando Desempenhar Esta Tarefa

Uma instalação silenciosa utiliza o mesmo programa que a versão de interface gráfica com o usuário (GUI) utiliza. Entretanto, no lugar de exibir uma interface de assistente, a instalação silenciosa lê todas as suas respostas de um arquivo que você customiza ou dos parâmetros que você transmite para a linha de comandos. Consulte o arquivo wxssetup.response.txt para obter um arquivo de resposta de exemplo e uma descrição de cada opção.

1. Opcional: Se você escolher instalar o eXtreme Scale usando um arquivo de resposta, primeiro customize o arquivo.

   Nota: É necessário especificar o nome completo do arquivo de resposta. Especificar o caminho relativo faz com que a instalação falhe sem nenhuma indicação de que ocorreu um erro.

   a. Copie o arquivo de resposta do DVD do produto em sua unidade de disco.
   b. Abra e edite o arquivo de resposta no editor de texto de sua escolha. É necessário especificar os seguintes parâmetros:
      - O contrato de licença
      - O diretório de instalação

      Dica: Ao instalar o eXtreme Scale em um ambiente do WebSphere Application Server, o instalador usará o diretório de instalação para determinar onde a instância existente do WebSphere Application Server será instalada. Se você instalar em um nó que contenha várias instâncias do WebSphere Application Server, defina claramente o seu local.
   c. Execute o seguinte script para iniciar a instalação.

```
./install.sh|bat -options C:/drive_path/response_file.txt -silent
```

2. Opcional: Se você escolher instalar o eXtreme Scale ao transmitir determinados parâmetros para a linha de comandos, execute o seguinte script para iniciar a instalação:

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

Parâmetros de Instalação

Especifique os parâmetros na linha de comandos para customizar e configurar a instalação do produto.

Nota: É necessário especificar o nome completo do arquivo de resposta. Especificar o caminho relativo faz com que a instalação falhe sem nenhuma indicação de que ocorreu um erro.

Parâmetros

É possível transmitir os seguintes parâmetros durante a instalação da linha de comandos ou do arquivo de opções do produto:

-silent
Suprime a interface gráfica com o usuário (GUI). Especifique o parâmetro **-options** para indicar que o instalador conclui a instalação de acordo com um arquivo de opções customizadas. Se você não especificar o parâmetro **-options**, os valores padrão serão usados no lugar.

Exemplo de Uso

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

-options *path_name/file_name*
Especifica um arquivo de opções usado pelo instalador para executar uma instalação silenciosa. As propriedades na linha de comandos têm precedência.

Exemplo de Uso

```
./install.sh|bat -options c:/path_name/options_file.txt
```

-log #!file_name *@event_type*
Gera um arquivo de log de instalação que registra os seguintes tipos de eventos:

- err
- wrn
- msg1
- msg2
- dbg
- TODOS

Exemplo de Uso

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

-is:log *path_name/file_name*
Cria um arquivo de log que contém as procuras do Java Virtual Machine (JVM) do instalador enquanto tenta iniciar a GUI. O arquivo de log não é criado, a menos que seja especificado.

Exemplo de Uso

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

-is:javaconsole
Exibe uma janela do console durante o processo de instalação.

Exemplo de Uso

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

-is:silent
Suprime a janela de inicialização Java que normalmente é exibida quando o instalador é iniciado.

Exemplo de Uso

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

-is:tempdir *path_name*
Especifica o diretório temporário que o instalador usa durante a instalação.

Exemplo de Uso

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

Usando o Update Installer para instalar os pacotes de manutenção

Use o IBM Update Installer para atualizar o ambiente do WebSphere eXtreme Scale com vários tipos de manutenção, como correções temporárias, fix packs e pacotes de atualização.

Por Que e Quando Desempenhar Esta Tarefa

Use o IBM Update Installer para instalar e aplicar vários tipos de pacotes de manutenção para o WebSphere eXtreme Scale. Como o Update Installer executa manutenção regular, é necessário usar a versão mais atual da ferramenta.

1. Pare todos os processos que estão em execução no seu ambiente.
   - Para parar todos os processos que estão em execução no seu ambiente do eXtreme Scale independente, consulte o "Parando Servidores eXtreme Scale Independentes" na página 228 para obter mais informações.
   - Para parar todos os processos que estão em execução no seu ambiente do WebSphere Application Server independente, consulte Utilitários da Linha de Comandos .
2. Faça download da versão mais recente do Update Installer. Consulte Correções recomendadas para obter informações adicionais.
3. Instale o Update Installer. Consulte Instalando o Update Installer para WebSphere Software no Centro de Informações do WebSphere Application Server para obter informações adicionais.
4. Faça download no diretório *updi_root*/maintenance dos pacotes de manutenção que deseja instalar. Consulte o Site de Suporte para obter mais informações.
5. Utilize o Update Installer para instalar a correção temporária, fix pack ou pacote de atualizações. É possível instalar o pacote de manutenção executando a interface gráfica com o usuário (GUI) ou executando o Update Installer no modo silencioso.

   Execute o seguinte comando a partir do diretório *updi_root* para iniciar a GUI:

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

   Execute o seguinte comando a partir do diretório *updi_root* para executar o Update Installer no modo silencioso:

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

Se o processo de instalação falhar, consulte o arquivo de log temporário, que está no diretório *updi_root*/logs/update/tmp. O Update Installer cria o diretório *install_root*/logs/update/*maintenance_package*.install no qual os arquivos de log de instalação estão localizados.

Configurando um Object Request Broker Customizado

É possível usar uma versão customizada do Object Request Broker (ORB) com o WebSphere eXtreme Scale ao executar processos do Java Platform, Standard Edition independente no seu ambiente.

Antes de Iniciar

O WebSphere eXtreme Scale e o WebSphere Application Server fornecem um ORB, que já está configurado para uso com o eXtreme Scale. Em circunstâncias normais, não é necessário configurar o ORB ou usar um ORB diferente.

Por Que e Quando Desempenhar Esta Tarefa

O WebSphere eXtreme Scale usa o Object Request Broker (ORB) para ativar a comunicação entre os processos. Um ORB é incluído com o eXtreme Scale e o WebSphere Application Server. Se você estiver usando um IBM Developer Kit ou um SDK que seja fornecido com o WebSphere Application Server, o ORB será incluído no JRE.

É possível usar o ORB que é fornecido com o eXtreme Scale, com o IBM SDK ou o com o WebSphere Application Server. Quaisquer problemas que você tiver ao usar os ORBs a partir de fornecedores de software independentes deverão ser reproduzíveis com o IBM ORB e compatíveis com o JRE antes de entrar em contato com o suporte. O eXtreme Scale não suporta o ORB que é fornecido com o Sun Microsystems Java Development Kit (JDK). Enquanto o eXtreme Scale suporta kits do desenvolvedor a partir da maioria dos fornecedores, é recomendado usar o ORB que seja fornecido com o eXtreme Scale.

- Se o seu ambiente contiver um SDK Versão 5 ou posterior,atualize os scripts que iniciam o comando Java ao especificar um diretório alternativo.
  1. Copie os arquivos ibmorb.jar e ibmorbapi.jar customizados para um diretório vazio.
  - Conclua a seguinte etapa ao usar os scripts de produto em um ambiente eXtreme Scale independente:
    1. Edite o caminho para a variável OBJECTGRID_ENDORSED_DIRS no arquivo setupCmdLine para fazer referência ao diretório ORB customizado. Salve as alterações.

       Edite o arquivo *objectgridRoot*/bin/setupCmdLine.sh.

       Edite o arquivo *objectgridRoot*\bin\setupCmdLine.bat.
  - Conclua a seguinte etapa ao usar os scripts de produto em um ambiente WebSphere Application Server:
    1. Inclua a seguinte propriedade de sistema e os parâmetros no script startOgServer:

       ```
       -jvmArgs –Djava.endorsed.dirs=custom_ORB_directory
       ```
  - Conclua a seguinte etapa ao usar um script customizado para iniciar um processo de aplicativo cliente ou um processo de servidor:
    1. Inclua a seguinte propriedade de sistema no script customizado:

       ```
       -Djava.endorsed.dirs=custom_ORB_directory
       ```

- Se o seu ambiente contiver um SDK Versão 1.4.2, integre o ORB IBM no SDK especificado.
  1. Faça download e extraia o ORB a partir de um IBM SDK. Se nenhum IBM SDK estiver disponível para sua plataforma, faça download e extraia o IBM Developer Kit para Linux, Java Technology Edition. Consulte o IBM Developer Kits para obter mais informações.
  2. Copie os arquivos `java/jre/lib/ibmorb.jar` e `java/jre/lib/ibmorbapi.jar` para o diretório `java/jre/lib/ext` no SDK de destino.
  3. Crie ou edite o arquivo `orb.properties`, que está no diretório `java/jre/lib` do SDK. Inclua as seguintes propriedades ou verifique se elas existem no arquivo:

     ```
     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
     ```

     Para obter descrições das propriedades e configurações, consulte "Arquivo de Propriedades ORB" na página 197.
  4. Faça Download do Xerces2 Java 2.9. Consulte O Projeto Apache Xerces - Downloads para obter mais informações.
  5. Copie os arquivos `xercesImpl.jar` e `xml-apis.jar` para o diretório `lib/ext`.

Resultados

É possível usar o ORB customizado com o eXtreme Scale ao executar processos do cliente e do servidor doJava SE independente.

Desinstalando o WebSphere eXtreme Scale

Para remover o WebSphere eXtreme Scale do seu ambiente, é possível usar o assistente ou desinstalar silenciosamente o produto.

Antes de Iniciar

Atenção: O desinstalador remove todos os arquivos binários e toda a manutenção, como fix packs e correções temporárias, ao mesmo tempo.

1. Pare todos os processos que executam o eXtreme Scale. Caso contrário, a desinstalação falhará.
   - Se você instalou o eXtreme Scale independente, consulte o "Parando Servidores eXtreme Scale Independentes" na página 228.
   - Se você instalou o eXtreme Scale com uma instalação existente do WebSphere Application Server, consulteUtilitários de Linha de Comandos para obter mais informações sobre como parar os processos do WebSphere Application Server.
2. Execute o seguinte script:
   - UNIX Linux `install_root/uninstall_wxs/uninstall.`
   - Windows `install_root\uninstall_wxs\uninstall.exe`

Resultados

Você removeu o eXtreme Scale do seu ambiente.

Capítulo 5. Customizando o WebSphere eXtreme Scale for z/OS

Usando o WebSphere Customization Tools, é possível gerar e executar tarefas customizadas para customizar o WebSphere eXtreme Scale para z/OS.

Antes de Iniciar

- Verifique se o seu sistema contém o nível mais recente do WebSphere Application Server Network Deployment:
  - Se você estiver executando a Versão 6.1, o sistema deverá conter, pelo menos, um Fix Pack 27. Consulte Instalando o ambiente de serviço de aplicativo da Versão 6.1 para obter mais informações.
  - Se você estiver executando a Versão 7.0, o sistema deverá conter, pelo menos, um Fix Pack 3. Consulte Instalando o ambiente de serviço de aplicativo da Versão 7.0 para obter mais informações.
- Instale o WebSphere eXtreme Scale para z/OS. Consulte o Diretório do Programa do WebSphere eXtreme Scale na Página da Biblioteca para obter mais informações.

Por Que e Quando Desempenhar Esta Tarefa

Usando o WebSphere Customization Tools, gere definições de customização e faça upload e execute tarefas customizadas para customizar o WebSphere eXtreme Scale para z/OS.Consulte os tópicos a seguir para obter informações adicionais:

- "Instalando o WebSphere Customization Tools"
- "Gerando Definições de Customização" na página 56
- "Fazendo Upload e Executando Tarefas Customizadas" na página 57

Instalando o WebSphere Customization Tools

Instale o WebSphere Customization Tools Versão 7.0.0.3 ou superior para customizar o ambiente do WebSphere eXtreme Scale para z/OS.

Antes de Iniciar

Instale o WebSphere eXtreme Scale para z/OS. Consulte o Diretório do Programa do WebSphere eXtreme Scale na Página da Biblioteca para obter mais informações.

Por Que e Quando Desempenhar Esta Tarefa

O WebSphere Customization Tools é uma ferramenta gráfica baseada na estação de trabalho usada para criar tarefas customizadas que criam os ambientes de tempo de execução do WebSphere eXtreme Scale para z/OS.

1. Use o FTP para copiar os arquivos de extensão `xs.wct` e `xspf.wct` a partir do sistema z/OS para a estação de trabalho na qual você está instalando o WebSphere Customization Tools. Os arquivos de extensão estão no diretório `/usr/lpp/zWebSphereXS/util/V7R0/WCT` no sistema z/OS.
2. Faça download e instale o WebSphere Customization Tools Versão 7.0.0.3 ou superior a partir do Web site apropriado:
   - Windows WebSphere Customization Tools para Windows

- Linux WebSphere Customization Tools para Linux

3. Transfira por upload o arquivo `xs.wct` para o aplicativoWebSphere Customization Tools.
   a. Inicie o aplicativo WebSphere Customization Tools na sua estação de trabalho.
   b. Clique em **Ajuda** → **Atualizações de Software** → **Instalar Extensão**.
   c. No painel Locais de Extensão do WebSphere Customization Tools, clique em **Instalar novo local de extensão**.
   d. No painel Arquivo Archive de Origem, clique em **Procurar**, navegue para o diretório no qual você copiou o arquivo `xs.wct` na etapa 1 e clique em **Abrir**.
   e. Clique em **Avançar** no painel Resumo.

      Nota: O painel Instalação com Êxito é exibido. Antes de clicar em **Concluir**, é necessário copiar e salvar os dados no campo de local:

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

   f. No painel Configuração do Produto, clique em **Incluir um Local de Extensão**. Cole os dados copiados na etapa anterior no campo Local e clique em **OK**.
   g. Clique em **Sim** para reiniciar o WebSphere Customization Tools.
4. Transfira por upload o arquivo `xspf.wct` para o aplicativo WebSphere Customization Tools.
   a. Clique em **Ajuda** → **Atualizações de Software** → **Instalar Extensão**.
   b. No painel Locais de Extensão do WebSphere Customization Tools, clique em **Instalar novo local de extensão**.
   c. No painel Arquivo Archive de Origem, clique em **Procurar**, navegue para o diretório no qual você copiou o arquivo `xspf.wct` na etapa 1 e clique em **Abrir**.
   d. Clique em **Avançar** no painel Resumo.

      Nota: O painel Instalação com Êxito é exibido. Antes de clicar em **Concluir**, é necessário copiar e salvar os dados do campo de local:

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

   e. No painel Configuração do Produto, clique em **Incluir um Local de Extensão**. Cole os dados copiados na etapa anterior no campo Local e clique em **OK**.
   f. Clique em **Sim** para reiniciar o WebSphere Customization Tools.

O que Fazer Depois

Depois de fazer upload dos dois arquivos de extensão e de reiniciar o WebSphere Customization Tools, use a ferramenta de Gerenciamento de Perfil para gerar definições de customização para o eXtreme Scale para z/OS. Consulte o “Gerando Definições de Customização” para obter informações adicionais.

Gerando Definições de Customização

Use a função da Profile Management Tool no WebSphere Customization Tools para gerar definições de customização e criar tarefas customizadas para o WebSphere eXtreme Scale para z/OS.

Antes de Iniciar

Instale o WebSphere Customization Tools e faça upload dos arquivos de extensão `xs.wct` e `xspf.wct`. Consulte o "Instalando o WebSphere Customization Tools" na página 55 para obter informações adicionais.

Por Que e Quando Desempenhar Esta Tarefa

É possível gerar definições de customização usando a Profile Management Tool, que é fornecido no WebSphere Customization Tools. Uma *definição de customização* é um conjunto de arquivos usados para criar tarefas customizadas com o propósito de configurar o WebSphere eXtreme Scale para z/OS.

1. Inicie a Ferramenta Gerenciamento de Perfis.
   - Windows Clique em **Iniciar** → **Programas** → **IBM WebSphere** → **WebSphere Customization Tools**. Depois que o aplicativo for iniciado, clique na guia **Profile Management Tool**.
   - Linux Clique em *operating_system_menus* → **IBM WebSphere** → **WebSphere Customization Tools**. Depois que o aplicativo for iniciado, clique na guia **Profile Management Tool**.
2. Inclua um local existente ou crie um novo local da definição de customização que deseja criar. Na guia **Locais de Customização**, clique em **Incluir**. Se você criar um novo local, a caixa Versão fará referência à versão existente do produtoWebSphere Application Server instalada no sistema z/OS.

   Nota: Não use o mesmo local que está sendo usado para outras definições de customização do eXtreme Scale.
3. Gere a definição de customização. Na guia **Definições de Customização**, clique em **Aumentar**.
4. Selecione o tipo de ambiente de definição a ser criado:
   - Nó do Servidor de Aplicativos Independente
   - Gerenciador de implementação
   - Servidor de Aplicativos
   - Nó gerenciado (customizado)
5. Preencha os campos nos painéis. Especifique os valores dos parâmetros que são usados para criar o sistema z/OS.
6. Clique em **Aumentar** para gerar a definição de customização.

O que Fazer Depois

Faça upload da tarefa customizada para o sistema z/OS de destino. Consulte o "Fazendo Upload e Executando Tarefas Customizadas" para obter informações adicionais.

Fazendo Upload e Executando Tarefas Customizadas

Depois de gerar as definições de customização, você poderá fazer upload e executar as tarefas customizadas associadas às definições do seu sistema WebSphere eXtreme Scale para z/OS.

Antes de Iniciar

Crie as definições de customização para as tarefas que você deseja transferir por upload para um sistema z/OS. Consulte o “Gerando Definições de Customização” na página 56 para obter informações adicionais.

Por Que e Quando Desempenhar Esta Tarefa

Faça upload e execute as tarefas customizadas criadas usando o WebSphere Customization Tools para administrar e monitorar seu ambiente WebSphere eXtreme Scale para z/OS.

1. Fazer upload das tarefas customizadas. Na guia **Definições de Customização**, selecione as tarefas que deseja fazer upload e clique em **Processar**.
2. Faça upload das tarefas para o servidor de FTP no seu sistema z/OS. Especifique as informações necessárias no painel **Fazer Upload da Definição de Customização**.
3. Clique em **Concluir**.
4. Execute as tarefas customizadas. Clique na guia **Instruções de Customização** e siga as instruções de customização para cada tarefa.

Capítulo 6. Configurando WebSphere eXtreme Scale

É possível configurar o WebSphere eXtreme Scale para executar em um ambiente independente ou configurar o eXtreme Scale para executar em um ambiente que contenha o WebSphere Application Server ou o WebSphere Application Server Network Deployment. Para que uma implementação do eXtreme Scale selecione alterações de configuração no lado da grade do servidor, será necessário reiniciar os processos para que essas alterações tomem efeito em vez de serem aplicadas dinamicamente. Porém, no lado do cliente, embora não seja possível alterar as definições de configuração para uma instância de cliente existente, você poderá criar um novo cliente com as configurações necessárias ao usar um arquivo XML ou fazer isso programaticamente. Ao criar um cliente, será possível substituir as configurações padrão fornecidas com a configuração do servidor atual.

Configurando um ObjectGrid Local de Memória

Uma configuração do eXtreme Scale local de memória pode ser criada usando um arquivo XML descritor do ObjectGrid ou as APIs do eXtreme Scale.

Por Que e Quando Desempenhar Esta Tarefa

Segue um exemplo simples de XML: o arquivo `companyGrid.xml`. As primeiras linhas do arquivo incluem o cabeçalho obrigatório de cada arquivo XML do ObjectGrid. O arquivo define o CompanyGrid ObjectGrid com o Customer, Item, OrderLine e Order BackingMaps. Um arquivo XML de política de implementação é passado para um contêiner eXtreme Scale durante a inicialização.

arquivo 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>
```

Transmita o arquivo XML para um dos métodos na createObjectGrid na interface ObjectGridManager. O seguinte código de amostra valida o arquivo `companyGrid.xml` em relação ao esquema XML e cria o CompanyGrid do ObjectGrid. A instância do ObjectGrid recém-criada não é armazenada em cache.

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

Como alternativa para o uso do XML, os objetos do ObjectGrid podem ser criados programaticamente. O seguinte código de amostra pode ser utilizado no lugar do XML e do código anterior.

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

O eXtreme Scale possui vários plug-ins e atributos customizáveis. Para obter uma descrição completa do arquivo XML do ObjectGrid, consulte eXtreme Scale referência de configuração.

interface ObjectGrid

Os seguintes métodos permitem interagir com uma instância ObjectGrid.

Introdução

Criar e Inicializar

Consulte o tópico da interface ObjectGridManager para obter as etapas necessárias para criação de uma instância do ObjectGrid. Existem dois métodos distintos para criar uma instância do ObjectGrid: programaticamente ou com arquivos de configuração XML. Consulte a documentação da API para obter informações adicionais.

Métodos Get ou Set e Factory

Quaisquer métodos configurados devem ser chamados antes de inicializar a instância do ObjectGrid. Se você chamar o método set depois de o método initialize ser chamado, o resultado será uma exceção java.lang.IllegalStateException. Cada um dos métodos getSession da interface ObjectGrid também chama implicitamente o método initialize. Portanto, é necessário chamar os métodos set antes de chamar qualquer um dos métodos getSession. A única exceção desta regra é com a configuração, a inclusão e a remoção de objetos ObjectGridEventListener. Esses objetos podem ser processados após a conclusão do processo de inicialização.

A interface ObjectGrid contém os seguintes métodos principais.

Tabela 5. Métodos de Interface ObjectGrid

Método	Descrição
BackingMap defineMap(String name);	defineMap: é um método factory para definir um BackingMap exclusivamente denominado. Para obter informações adicionais sobre os mapas de apoio, consulte a Interface BackingMap.
BackingMap getMap(String name);	getMap: Retorna um BackingMap anteriormente definido chamando defineMap. Utilizando este método, é possível configurar o BackingMap, se ainda não estiver configurado por meio da configuração XML.
BackingMap createMap(String name);	createMap: Cria um BackingMap, mas não o armazena em cache para uso por este ObjectGrid. Use este método com o método setMaps(List) da interface ObjectGrid, que captura BackingMaps para uso com este ObjectGrid. Utilize estes métodos quando estiver configurando um ObjectGrid com Spring Framework.
void setMaps(List mapList);	setMaps: Limpa quaisquer BackingMaps que foram anteriormente definidos neste ObjectGrid e os substitui pela lista de BackingMaps que é fornecida.
public Session getSession() throws ObjectGridException, TransactionCallbackException;	getSession: Retorna um Session, que fornece a funcionalidade begin, commit e rollback para uma Unidade de Trabalho. Para obter informações adicionais sobre os objetos Session, consulte a interface Session.
Session getSession(CredentialGenerator cg);	getSession(CredentialGenerator cg): Obtém uma sessão com um objeto CredentialGenerator. Este método pode ser chamado apenas pelo cliente do ObjectGrid em um ambiente do servidor do cliente.
Session getSession(Subject subject);	getSession(Subject subject): Permite o uso de um objeto Subject específico ao invés de um configurado no ObjectGrid para obter um Session.

Tabela 5. Métodos de Interface ObjectGrid (continuação)

Método	Descrição
void initialize() throws ObjectGridException;	initialize: O ObjectGrid é inicializado e está disponível para uso geral. Este método é chamado implicitamente quando o método getSession é chamado, se o ObjectGrid não estiver em um estado inicializado.
void destroy();	destroy: A estrutura é desmontada e não pode ser utilizada após este método ser chamado.
void setTxTimeout(int timeout);	setTxTimeout: Utilize este método para configurar a quantidade de tempo, em segundos, que uma transação que é iniciada por uma sessão que esta instância do ObjectGrid criou tem permissão para ser concluída. Se uma transação não for concluída dentro de uma quantidade de tempo especificada, a Sessão que iniciou a transação será marcada como "expirada". Uma Sessão marcada com tempo limite expirado faz com que o próximo método ObjectMap que for chamado pela Sessão marcada resulte em uma exceção . O objeto Session é marcado apenas como rollback, o que faz com que ocorra um retrocesso na transação mesmo se o aplicativo chamar o método commit ao invés do método rollback após a exceção TransactionTimeoutException ser capturada pelo aplicativo. Um valor de tempo limite de 0 indica que a transação tem permissão para uma quantidade de tempo ilimitada para ser concluída. A transação não expira se um valor do tempo limite de 0 for utilizado. Se este método não for chamado, então, qualquer objeto Session que é retornado pelo método getSession desta interface possui um valor de tempo limite configurado com 0, por padrão. Um aplicativo pode substituir a configuração de tempo limite da transação em uma base por Sessão utilizando o método setTransactionTimeout da interface com.ibm.websphere.objectgrid.Session. Você também pode configurar o tempo limite da transação com o arquivo `objectGrid.xml` no caso distribuído.
int getTxTimeout();	getTxTimeout: Retorna o valor de tempo limite da transação em segundos. Este método retorna o mesmo valor que foi transmitido como o parâmetro de tempo limite no método setTxTimeout. Se o método setTxTimeout não foi chamado, então, o método retorna 0 para indicar que a transação tem permissão para uma quantidade de tempo ilimitada para ser concluída.
public int getObjectGridType();	Retorna o tipo de ObjectGrid. O valor de retorno é equivalente a uma das constantes declaradas nesta interface: LOCAL, SERVER ou CLIENT.
public void registerEntities(Class[] entities);	Registre uma ou mais entidades com base nos metadados da classe. O registro da entidade é necessário antes da inicialização do ObjectGrid para ligar uma Entidade a um BackingMap e qualquer índice definido. Esse método pode ser chamado várias vezes.
public void registerEntities(URL entityXML);	Registra uma ou mais entidades de um arquivo XML da entidade. O registro da entidade é necessário antes da inicialização do ObjectGrid para ligar uma Entidade a um BackingMap e qualquer índice definido. Esse método pode ser chamado várias vezes.
void setQueryConfig(QueryConfig queryConfig);	Configure o objeto QueryConfig para este ObjectGrid. Um objeto QueryConfig fornece configurações de consulta para executar consultas de objeto nos mapas nesse ObjectGrid.
//Palavras-chave	
void associateKeyword(Serializable parent, Serializable child);	Reprovado: Utilize o Índice ou a função de consulta para obter Objects com atributos específicos. O método associateKeyword fornece um mecanismo de invalidação flexível baseado em palavras-chave. Este método vincula as duas palavras-chave em um relacionamento direcional. Se o pai for invalidado, o filho também será invalidado. A invalidação do filho não tem nenhum impacto sobre o pai.
//Segurança	
void setSecurityEnabled()	setSecurityEnabled: Ativa a segurança. A segurança é desativada por padrão.
void setPermissionCheckPeriod(long period);	setPermissionCheckPeriod: Este método obtém um parâmetro único que indica com que frequência verificar a permissão que é utilizada para permitir o acesso de um cliente. Se o parâmetro for 0, todos os métodos solicitam ao mecanismo de autorização, autorização JAAS ou autorização customizada, para verificar se o objeto atual tem permissão. Esta estratégia pode causar problemas de desempenho, dependendo da implementação de autorização. No entanto, este tipo de autorização está disponível se for requerido. Alternativamente, se o parâmetro for menor do que 0, ele indica o número de milissegundos a armazenar um conjunto de permissões em cache antes de retornar para o mecanismo de autorização para atualizá-las. Este parâmetro fornece um desempenho muito melhor mas, se as permissões de backend forem alteradas durante este período, o ObjectGrid poderá permitir ou impedir o acesso mesmo que o provedor de segurança de backend tenha sido modificado.
void setAuthorizationMechanism(int authMechanism);	setAuthorizationMechanism: Configurar o mecanismo de autorização. O padrão é SecurityConstants.JAAS_AUTHORIZATION.

Tabela 5. Métodos de Interface ObjectGrid (continuação)

Método	Descrição
setMapAuthorization(MapAuthorization ma);	Reprovado: Use o método setObjectGridAuthorization (ObjectGridAuthorization) em vez do plug-in de autorizações customizadas. setMapAuthorization: Configura o plug-in MapAuthorization para esta instância do ObjectGrid. Este plug-in pode ser utilizado para autorizar acessos de ObjectMap ou JavaMap aos proprietários que estão contidos no objeto Subject. Uma implementação típica deste plug-in é recuperar os proprietários do objeto Subject e, em seguida, verificar se as permissões especificadas são concedidas aos proprietários.
setSubjectSource(SubjectSource ss);	setSubjectSource: Configura o plug-in SubjectSource. Este plug-in pode ser utilizado para obter um objeto Subject que representa o cliente do ObjectGrid. Este subject é utilizado para autorização do ObjectGrid. O método SubjectSource.getSubject é chamado pelo tempo de execução do ObjectGrid quando o método ObjectGrid.getSession é utilizado para obter uma sessão e a segurança está ativada. Este plug-in é útil para um cliente já autenticado: ele pode recuperar o objeto Subject autenticado e, em seguida, transmitir para a instância do ObjectGrid. Outra autenticação não é necessária.
setSubjectValidation(SubjectValidation sv);	setSubjectValidation: Configura o plug-in SubjectValidation para esta instância do ObjectGrid. Este plug-in pode ser utilizado para validar se um subject javax.security.auth.Subject transmitido para o ObjectGrid é um subject válido que não foi violado. Uma implementação deste plug-in precisa de suporte do criador do objeto Subject, porque apenas o criador sabe se o objeto Subject foi violado. No entanto, um criador do subject pode não saber se o Subject foi violado. Neste caso, este plug-in não deve ser utilizado.
void setObjectGridAuthorization(ObjectGridAuthorization ogAuthorization);	Configura o ObjectGridAuthorization para esta instância do ObjectGrid. Transmitir nulo para este método remove um objeto ObjectGridAuthorization configurado anteriormente de uma chamada anterior deste método e indica que <code>ObjectGrid</code> não está associado a um objeto ObjectGridAuthorization. Esse método deve ser utilizado somente quando a segurança do ObjectGrid estiver ativada. Se a segurança do ObjectGrid estiver desativada, o objeto ObjectGridAuthorization não será utilizado. Um plug-in do ObjectGridAuthorization pode ser utilizado para autorizar o acesso ao ObjectGrid e aos mapas. A partir do XD 6.1, o setMapAuthorization é reprovado e o setObjectGridAuthorization é recomendado para uso. Entretanto, se o plug-in de MapAuthorization e o plug-in de ObjectGridAuthorization forem utilizados, o ObjectGrid utilizará o MapAuthorization fornecido para autorizar os acessos ao mapa, mesmo que seja reprovado.

interface ObjectGrid: Plug-ins

A interface ObjectGrid possui diversos pontos de plug-in opcionais para interações mais extensíveis.

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

- ObjectGridEventListener: Uma interface ObjectGridEventListener é utilizada para receber notificações quando ocorrem eventos significativos no ObjectGrid. Estes eventos incluem inicialização do ObjectGrid, início de uma transação, encerramento de uma transação e destruição de um ObjectGrid. Para atender estes eventos, crie uma classe que implementa a interface ObjectGridEventListener e inclua-a no ObjectGrid. Estes listeners estão associados a cada Sessão. Consulte Interface Listeners e Session para obter informações adicionais.
- TransactionCallback: Uma interface listener TransactionCallback permite que eventos transacionais, tais como sinais begin, commit e rollback, para envia para esta interface. Geralmente, uma interface listener TransactionCallback é utilizada com um Utilitário de Carga. Para obter informações adicionais, consulte o plug-in deTransactionCallback e os Utilitários de Carga. Estes eventos podem então ser utilizados para coordenar transações com um recurso externo ou em vários utilitários de carga.

- reserveSlot: Permite que plug-ins neste ObjectGrid reservem slots para uso em instâncias do objeto que possuem slots como TxID.
- SubjectValidation. Se a segurança estiver ativada, este plug-in pode ser utilizado para validar uma classe javax.security.auth.Subject que é passada para o ObjectGrid.
- MapAuthorization. Se a segurança estiver ativada, este plug-in poderá ser utilizado para autorizar acessos do ObjectMap aos proprietários representados pelo objeto Subject.
- SubjectSource: Se a segurança estiver ativada, este plug-in pode ser utilizado para obter um objeto Subject que representa o cliente do ObjectGrid. Este subject é então utilizado para autorização do ObjectGrid.

Para obter informações adicionais sobre plug-ins, consulte a introdução aos plug-ins no *Guia de Programação*.

Interface BackingMap

Cada instância do ObjectGrid contém uma coleta de objetos de BackingMap. Use o método defineMap ou o método createMap da interface ObjectGrid para nomear e incluir cada BackingMap em uma instância do ObjectGrid. Estes métodos retornam uma instância do BackingMap que é então utilizada para definir o comportamento de um Mapa individual.

Interface Session

A interface Session é usada para iniciar uma transação e obter o ObjectMap ou JavaMap que é necessário para execução de interação transacional entre um aplicativo e um objeto BackingMap. No entanto, as alterações na transação não serão aplicadas ao objeto de BackingMap até que a transação seja confirmada. Um BackingMap pode ser considerado como um cache de memória de dados com commit para um mapa individual. Para obter informações adicionais, consulte as informações sobre o uso de Sessões para acesso a dados no *Guia de Programação*.

A interface BackingMap oferece métodos para configuração de atributos do BackingMap. Alguns dos métodos set permitem a extensibilidade de um BackingMap por meio de vários plug-ins projetados customizados. Consulte a lista dos métodos de configuração a seguir para configurar atributos e fornece suporte a plug-ins desenvolvidos de forma customizada:

```
// For setting BackingMap attributes.
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);

// For setting an optional custom plug-in provided by application.
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);
```

Existe um método get correspondente para cada um dos métodos set listados.

Atributos de BackingMap

Cada BackingMap possui os seguintes atributos que podem ser configurados para modificar ou controlar o comportamento de BackingMap:

- ReadOnly: Este atributo indica se o Mapa é um Mapa somente leitura ou um Mapa de leitura e gravação. Se este atributo nunca for configurado para o Mapa, o Mapa será padronizado para ser um Mapa de leitura e gravação. Quando um BackingMap é configurado para ser de leitura, o ObjectGrid otimiza o desempenho para de leitura quando possível.
- NullValuesSupported: Este atributo indica se um valor nulo pode ser colocado no Mapa. Se este atributo nunca for configurado, o Mapa não suportará valores nulos. Se o Mapa suportar valores nulos, uma operação get que retorna nulo poderá significar que o valor é nulo ou o mapa não contém a chave especificada pela operação get.
- LockStrategy: Este atributo determina se um gerenciador de bloqueios é utilizado por este BackingMap. Se um gerenciador de bloqueios for utilizado, o atributo LockStrategy será utilizado para indicar se será utilizada uma abordagem de bloqueio otimista ou pessimista para bloquear as entradas do mapa. Se este atributo não for configurado, será utilizado o LockStrategy otimista. Consulte o tópico Bloqueio para obter detalhes sobre as estratégias de bloqueio suportadas.
- CopyMode: Este atributo determina se uma cópia de um objeto de valor é feita no BackingMap quando um valor é lido a partir do mapa ou é colocado no BackingMap durante o ciclo de commit de uma transação. Vários modos de cópia são suportados para permitir que o aplicativo faça o equilíbrio entre desempenho e integridade de dados. Se este atributo não estiver configurado, então, o modo de cópia COPY_ON_READ_AND_COMMIT é utilizado. Este modo de cópia não tem o melhor desempenho, mas tem a maior proteção contra problemas de integridade de dados. Se o BackingMap estiver associado a uma entidade da API EntityManager, então a configuração CopyMode não tem efeito a menos que o valor esteja configurado como COPY_TO_BYTES. Se qualquer outro CopyMode estiver configurado, ele sempre será configurado como NO_COPY. Para substituir o CopyMode para uma mapa de entidades, use o método ObjectMap.setCopyMode. Para obter informações mais detalhadas sobre os modos de cópia, consulte as informações sobre as boas práticas do método CopyMode no *Guia de Programação*.
- CopyKey: Este atributo determina se o BackingMap faz uma cópia de um objeto de chave quando uma entrada é criada pela primeira vez no mapa. A ação padrão é não fazer uma cópia de objetos de chave, porque as chaves normalmente são objetos inalteráveis.
- NumberOfBuckets: Este atributo indica o número de depósitos de hash a serem utilizados pelo BackingMap. A implementação de BackingMap utiliza um mapa hash para sua implementação. Se existirem muitas entradas no BackingMap, mais depósitos significam melhor desempenho. O número de chaves que possuem o mesmo depósito se torna mais baixo conforme aumenta o número de depósitos. Mais depósitos também significam mais simultaneidade. Este atributo é útil para ajuste de desempenho. Um valor padrão indefinido será utilizado se o aplicativo não configurar o atributo NumberOfBuckets.
- NumberOfLockBuckets: Este atributo indica o número de depósitos de bloqueio que são utilizados pelo gerenciador de bloqueios para este BackingMap. Quando LockStrategy estiver configurado como OPTIMISTIC ou PESSIMISTIC, será criado um gerenciador de bloqueios para o BackingMap. O gerenciador de

bloqueios utiliza um mapa hash para rastrear as entradas que estão bloqueadas por uma ou mais transações. Se existirem muitas entradas no mapa de hash, mais depósitos de bloqueios resultarão em melhor desempenho, porque o número de chaves que colidem no mesmo depósito é mais baixo conforme aumenta o número de depósitos. Mais depósitos de bloqueios também significam mais simultaneidade. Quando o atributo LockStrategy for configurado como NONE, nenhum gerenciador de bloqueios será utilizado por este BackingMap. Neste caso, a configuração de numberOfLockBuckets não tem nenhum efeito. Se este atributo não for configurado, será utilizado um valor indefinido.

- LockTimeout: Este atributo é utilizando quando o BackingMap está utilizando um gerenciador de bloqueios. O BackingMap utilizará um gerenciador de bloqueios quando o atributo LockStrategy for configurado como OPTIMISTIC ou PESSIMISTIC. O valor de atributo está em segundos e determina por quanto tempo o gerenciador de bloqueios espera que um bloqueio seja concedido. Se este atributo não estiver configurado, então são usados 15 segundos como o valor de LockTimeout. Consulte Bloqueio Pessimista para obter detalhes sobre as exceções de tempo limite de espera de bloqueio que podem ocorrer.
- TtlEvictorType: Cada BackingMap possui seu próprio evictor time to live integrado que utilize um algoritmo baseado em tempo para determinar quais entradas de mapa despejar. Por padrão, o evictor time to live integrado não está ativo. É possível ativar o evictor time to live chamando o método setTtlEvictorType com um dos três valores: CREATION_TIME, LAST_ACCESS_TIME ou NONE. Um valor de CREATION_TIME indica que o evictor inclui o atributo TimeToLive no horário em que a entrada de mapa foi criada no BackingMap para determinar quando o evictor deve despejar a entrada de mapa do BackingMap. Um valor de LAST_ACCESS_TIME indica que o evictor inclui o atributo TimeToLive no horário que a entrada de mapa foi acessada pela última vez por alguma transação que o aplicativo está executando para determinar quando o evictor deve despejar a entrada de mapa. A entrada do mapa será liberada apenas se uma entrada de mapa nunca for acessada por nenhuma transação durante um período de tempo especificado pelo atributo TimeToLive. Um valor de NONE indica que o evictor deve permanecer inativo e nunca liberar nenhuma das entradas de mapa. Se este atributo nunca for configurado, NONE será utilizado como o padrão e o evictor time to live não será ativado. Consulte Evictors para obter detalhes sobre o evictor time to live interno.
- TimeToLive: Este atributo é utilizado para especificar o número de segundos que o evictor time to live integrado precisa incluir no horário da criação ou no horário do último acesso para cada entrada, conforme descrito para o atributo TtlEvictorType. Se este atributo nunca for configurado, será utilizado o valor especial de zero para indicar que o time to live é infinito. Se este atributo for configurado como infinito, as entradas do mapa nunca serão liberadas pelo evictor.

O exemplo a seguir demonstra como definir o someMap BackingMap na instância someGrid ObjectGrid e configura diversos atributos do BackingMap utilizando os métodos set da interface BackingMap:

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

...

ObjectGrid og =
ObjectGridManagerFactory.getObjectGridManager().createObjectGrid("someGrid");
BackingMap bm = objectGrid.getMap("someMap");
bm.setReadOnly( true ); // override default of read/write
```

```
bm.setNullValuesSupported(false); // override default of allowing Null values
bm.setLockStrategy( LockStrategy.PESSIMISTIC ); // override default of OPTIMISTIC
bm.setLockTimeout( 60 ); // override default of 15 seconds.
bm.setNumberOfBuckets(251); // override default (prime numbers work best)
bm.setNumberOfLockBuckets(251); // override default (prime numbers work best)
```

Plug-ins do BackingMap

A interface BackingMap possui vários pontos de conexão opcionais para interações mais extensíveis com o ObjectGrid:

- **Plug-in ObjectTransformer**: Para algumas operações do mapa, um BackingMap talvez precise serializar, desserializar ou copiar uma chave ou valor de uma entrada no BackingMap. O BackingMap pode desempenhar estas ações fornecendo uma implementação padrão da interface ObjectTransformer. Um aplicativo pode melhorar o desempenho fornecendo um plug-in ObjectTransformer projetado customizado que é utilizado pelo BackingMap para serializar, desserializar ou copiar uma chave ou valor de uma entrada no BackingMap. Consulte as informações sobre o plug-in ObjectTransformer no *Guia de Programação* para obter informações adicionais.
- **Plug-in do Evictor**: O evictor time to live interno utiliza um algoritmo baseado em tempo para decidir quando uma entrada no BackingMap deve ser liberada. Alguns aplicativos talvez precisem utilizar um algoritmo diferente para decidir quando uma entrada em um BackingMap precisa ser liberada. O plug-in do Evictor disponibiliza um Evictor projetado customizado para ser utilizado pelo BackingMap. O plug-in do Evictor é uma inclusão no evictor time to live interno. Ele não substitui o evictor time to live. O ObjectGrid fornece um plug-in do Evictor customizado que implementa algoritmos bem conhecidos, como "menos utilizado recentemente" ou "menos utilizado freqüentemente". Os aplicativos podem conectar um dos plug-ins do Evictor fornecidos ou podem fornecer seu próprio plug-in do Evictor. Consulte as informações sobre o despejo no *Guia de Programação*.
- **Plug-in MapEventListener**: Um aplicativo talvez queira saber sobre eventos de BackingMap, como uma evicção de entrada do mapa ou um pré-carregamento de uma conclusão de BackingMap. Um BackingMap chama métodos no plug-in MapEventListener para notificar um aplicativo sobre eventos do BackingMap. Um aplicativo pode receber notificação de vários eventos de BackingMap, utilizando o método setMapEventListener para fornecer um ou mais plug-ins MapEventListener projetados customizados para o BackingMap. O aplicativo pode modificar os objetos MapEventListener listados usando o método addMapEventListener ou o método removeMapEventListener. Consulte as informações sobre o plug-in MapEventListener no *Guia de Programação* para obter informações adicionais.
- **Plug-in do Loader**: Um BackingMap é um cache de memória de um Mapa. Um plug-in do Loader é uma opção utilizada pelo BackingMap para mover dados entre a memória e é utilizado para armazenamento persistente para o BackingMap. Por exemplo, um utilitário de carga JDBC (Java Database Connectivity) pode ser usado para mover dados para dentro e para fora de um BackingMap e uma ou mais tabelas relacionais de um banco de dados relacional. Um banco de dados relacional não precisa ser utilizado como o armazenamento persistente para um BackingMap. O utilitário de carga também pode ser usado para dados movidos entre um BackingMap e um arquivo, entre um BackingMap e um mapa Hibernate, entre um BackingMap e um bean de entidade Java 2 Platform, Enterprise Edition (J2EE), entre um BackingMap e outro servidor de aplicativos, e assim por diante. O aplicativo deve fornecer um plug-in do Loader projetado customizado para mover dados entre o BackingMap e o armazenamento persistente para cada tecnologia utilizada. Se um Loader não for

fornecido, o BackingMap se tornará um cache de memória simples. Consulte as informações sobre o uso de um utilitário de carga no *Guia de Programação* para obter informações adicionais.

- **Plug-in OptimisticCallback**: Quando o atributo LockStrategy para um BackingMap for configurado como OPTIMISTIC, o BackingMap ou um plug-in do Loader deverá desempenhar operações de comparação para os valores do mapa. O plug-in OptimisticCallback é utilizado pelo BackingMap e pelo Loader para desempenhar as operações de comparação de controle de versões otimistas. Consulte as informações sobre o plug-in OptimisticCallback no *Guia de Programação* para obter informações adicionais.
- **Plug-in MapIndexPlugin**: Um plug-in MapIndexPlugin ou, abreviando, um Index, é uma opção utilizada pelo BackingMap para construir um índice baseado no atributo especificado do objeto armazenado. O índice permite que o aplicativo localize objetos por um valor específico ou intervalo de valores. Existem dois tipos de índice: estático e dinâmico. Consulte Indexação para obter informações detalhadas.

Para obter informações adicionais sobre plug-ins, consulte a introdução aos plug-ins no *Guia de Programação*.

Bloqueio de Entrada de Mapa

Um BackingMap do ObjectGrid suporta diversas estratégias de bloquei para manter a consistência de entradas de cache.

Cada BackingMap pode ser configurado para utilizar uma das seguintes estratégias de bloqueio:

1. Modo de Bloqueio Otimista
2. Modo de Bloqueio Pessimista
3. Nenhum(a)

A estratégia de bloqueio padrão é OPTIMISTIC. Utilize o bloqueio optimistic quando os dados são alterados de maneira infrequente. Os bloqueios são mantidos apenas por uma curta duração enquanto os dados estão sendo lidos do cache e copiados para a transação. Quando o cache da transação é sincronizado com o cache principal, quaisquer objetos de cache que foram atualizados são verificados junto à versão original. Se a verificação falhar, então, ocorre o rollback da transação e o resultado é uma exceção OptimisticCollisionException.

A estratégia de bloqueio PESSIMISTIC adquire bloqueios para entradas de cache e deve ser utilizada quando os dados são alterados frequentemente. Sempre que uma entrada de cache é lida, um bloqueio é adquirido e mantido condicionalmente até que a transação seja concluída. A duração de alguns bloqueios pode ser ajustada utilizando níveis de isolamento de transação para a sessão.

Se o bloqueio não for necessário porque os dados nunca são atualizados ou são atualizados apenas durante períodos tranquilos, você pode desativar o bloqueio utilizando a estratégia de bloqueio NONE. Esta estratégia é muito rápida porque um gerenciador de bloqueio não é necessário. A estratégia de bloqueio NONE é ideal para tabelas de consulta ou mapas somente leitura.

Para obter mais informações sobre as estratégias de bloqueio, consulte as informações sobre as estratégias de bloqueio no *Visão Geral do Produto*.

Especificando uma Estratégia de Bloqueio

O exemplo a seguir demonstra como a estratégia de bloqueio pode ser configurada nos BackingMaps map1, map2 e map3, em que cada mapa está utilizando uma estratégia de bloqueio diferente. O primeiro fragmento mostra como usar o XML para a configuração de estratégia de bloqueio e o segundo fragmento mostra uma abordagem programática.

Abordagem XML

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

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

Abordagem Programática

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

Para evitar uma exceção java.lang.IllegalStateException, o método setLockStrategy deve ser chamado antes de usar os métodos initialize ou getSession na instância do ObjectGrid local.

Configuração do Gerenciador de Bloqueios

Quando a estratégia de bloqueio pessimistic ou optimistic for utilizada, será criado um gerenciador de bloqueios para o BackingMap. O gerenciador de bloqueios utiliza um mapa hash para rastrear as entradas que estão bloqueadas por uma ou mais transações. Se existirem muitas entradas de mapa no mapa hash, mais depósitos de bloqueio podem resultar em melhor desempenho. O risco de colisões de sincronização Java é inferior conforme a quantidade de depósitos aumenta. Mais depósitos de bloqueios também resultam em maior simultaneidade. Os exemplos acima mostram como um aplicativo pode configurar o número de depósitos de bloqueios a serem utilizados para um BackingMap especificado:

Para evitar uma exceção java.lang.IllegalStateException, o método setNumberOfLockBuckets deve ser chamado antes de chamar os métodos initialize ou getSession na instância do ObjectGrid. O parâmetro do método setNumberOfLockBuckets é um inteiro primitivo Java que especifica a quantidade de depósitos de bloqueio para uso. Utilizar um número primo permite uma distribuição uniforme de entradas do mapa sobre os depósitos de bloqueios. Um bom ponto de partida para obter melhor desempenho é configurar o número de

depósitos de bloqueios para aproximadamente dez por cento do número esperado de entradas do BackingMap.

Configurando uma Estratégia de Bloqueio

É possível definir uma estratégia otimista, pessimista ou de ausência de bloqueio na configuração do WebSphere eXtreme Scale.

Por Que e Quando Desempenhar Esta Tarefa

É possível especificar uma estratégia de bloqueio programaticamente ou com o XML. Para obter mais informações sobre o bloqueio, consulte as informações sobre as estratégias de bloqueio no *Visão Geral do Produto*.

- **Configure uma estratégia de bloqueio otimista**
  - Programaticamente

Especificar a estratégia otimista programaticamente
```
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 );
```

  - Usando XML

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

- **Configure uma estratégia de bloqueio pessimista**
  - Programaticamente

Especificar a estratégia pessimista programaticamente
```
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 );
```

  - Usando XML

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

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

- **Configure uma estratégia de ausência de bloqueio**
  - Programaticamente

Especificar uma estratégia de ausência de bloqueio programaticamente
```
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 );
```

– Usando XML

Especificar uma estratégia de ausência de bloqueio com XML

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

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

O que Fazer Depois

Para evitar uma exceção java.lang.IllegalStateException, é necessário chamar o método setLockStrategy antes de chamar os métodos initialize ou getSession na instância ObjectGrid.

Configurando uma Grade Distribuída do eXtreme Scale

Use o arquivo XML descritor de política de implementação para gerenciar sua topologia de implementação.

A política de implementação é codificada como um arquivo XML que é fornecido para o contêiner do eXtreme Scale. O arquivo XML especifica as seguintes informações:

- Os mapas que pertencem a cada conjunto de mapas.
- O número de partições
- O número de réplicas síncronas e assíncronas

Para obter informações sobre como iniciar os servidores de contêiner, consulte o “Iniciando Processos do Contêiner em um Ambiente WebSphere Application Server” na página 231 ou o “Iniciando Processos do Contêiner” na página 221.

A política de implementação também controla os seguintes comportamentos de colocação.

- O número mínimo de contêineres ativos antes da ocorrência da disposição
- A substituição automática de shards perdidos
- A disposição de cada shard de uma única partição em uma máquina diferente.

Para obter mais informações sobre o arquivo XML de política de implementação, consulte “Arquivo Descritor XML de Política de Implementação” na página 145.

As informações de terminal não estão pré-configuradas no ambiente dinâmico. Não há nomes de servidor ou informações de topologia física localizadas na política de implementação. Todos os shards em uma grade são automaticamente colocados em contêineres pelo serviço de catálogo. O serviço de catálogo usa as restrições que são definidas pela política de implementação para gerenciar automaticamente a colocação de shard. Essa colocação automática de shard facilita a configuração de grades maiores. Também é possível incluir servidores no seu ambiente, conforme necessário.

Nota: Em um ambiente do WebSphere Application Server, um tamanho de grupo principal de mais de 50 membros não é suportado.

Um arquivo XML de política de implementação é passado para um contêiner eXtreme Scale durante a inicialização. Uma política de implementação deve ser usada junto com o arquivo XML de ObjectGrid. A política de implementação não é requerida para iniciar um contêiner, mas é recomendada. A política de implementação deve ser compatível com o ObjectGrid XML utilizado com ela. Para cada elemento objectgridDeployment na política de implementação, você deve ter um elemento objectGrid correspondente no XML do seu ObjectGrid. Os mapas no objectgridDeployment devem ser consistentes com os backingMaps localizados no XML ObjectGrid. Cada backingMap deve ser referido dentro de um e apenas um mapSet.

No exemplo a seguir, o arquivo `companyGridDpReplication.xml` é destinado a formar um par com o arquivo `companyGrid.xml` correspondente.

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>
```

O arquivo `companyGridDpReplication.xml` possui um mapSet que é dividido em 11 partições. Cada partição deve ter exatamente uma réplica síncrona. O número de réplicas síncronas é dedicado pelos atributos minSyncReplicas e maxSyncReplicas. Como o atributo minSyncReplicas é configurado para 1, cada partição no mapSet deve ter pelo menos uma réplica síncrona disponível para processar transações de gravação. Como maxSyncReplicas é configurado para 1, cada partição deve ter no máximo 1 réplica síncrona. As partições nesse mapSet não possuem réplicas.

O atributo numInitialContainers instrui o serviço de catálogo para adiar a colocação até que quatro contêineres estejam disponíveis para suportar esse ObjectGrid. O atributo numInitialContainers será ignorado depois que o número especificado de contêineres for alcançado.

O arquivo `companyGridDpReplication.xml` demonstra uma maneira comum de configurar uma política de implementação, mas uma política de implementação pode oferecer até mesmo mais controle sobre como e quando seu ObjectGrid é implementado em seu ambiente. Para obter uma descrição completa do arquivo descritor de política de implementação, consulte “Arquivo Descritor XML de Política de Implementação” na página 145.

Topologia Distribuída

Caches consistentes distribuídos oferecem melhores desempenho, disponibilidade e escalabilidade que podem ser configurados.

O WebSphere eXtreme Scale equilibra automaticamente os servidores. É possível incluir servidores adicionais sem reiniciar o WebSphere eXtreme Scale. Incluir servidores adicionais sem precisar reiniciar o eXtreme Scale permite executar implementações simples e implementações que atingem terabytes em que milhares de servidores são necessários. Essa topologia de implementação é flexível. Com o serviço de catálogo, é possível incluir e remover servidores para melhorar o uso de recursos sem remover o cache inteiro. Para incluir ou remover um servidor, use o comando startOgServer para iniciar um servidor de contêiner, que se comunica com o serviço de catálogo com a opção **-catalogServiceEndPoints**. Todos os clientes de topologia distribuída se comunicam com o serviço de catálogo através do Internet Interoperability Object Protocol (IIOP). Todos os clientes utilizam a interface ObjectGrid para comunicar-se com servidores.

O recurso de configuração dinâmica do WebSphere eXtreme Scale facilita incluir recursos no sistema. Contêineres hospedam os dados e o serviço de catálogo funciona como o ponto de acesso para a grade. Os contêineres são responsáveis pela manutenção dos dados. O serviço de catálogo é responsável por encaminhar pedidos para o local correto no primeiro acesso, alocando espaço em contêineres de host e gerenciando o funcionamento e a disponibilidade geral do sistema. Os clientes se conectam a um serviço de catálogo, recuperam uma descrição da topologia de contêiner-servidor e, em seguida, comunica-se diretamente a cada servidor, conforme necessário. Quando a topologia do servidor é alterada porque novos servidores foram incluídos, ou em razão de falha de outros, o cliente é automaticamente redirecionado para o servidor apropriado que está hospedando os dados.

Um serviço de catálogo normalmente existe na sua própria grade do Java Virtual Machines. Um serviço de catálogo único pode ser utilizado para gerenciar vários servidores. Um contêiner pode ser iniciado em uma JVM por si ou pode ser carregado em uma JVM arbitrária com outros contêineres para diferentes servidores. Pode existir um cliente em alguma JVM e comunicar-se com um ou mais servidores. Um cliente também pode existir na mesma JVM como um contêiner.

Também é possível criar uma política de implementação programaticamente ao integrar um contêiner em um processo ou aplicativo Java existente. Para obter mais informações, consulte a API DeploymentPolicy do eXtreme Scale.

Configurando um Cliente do eXtreme Scale

É possível configurar um cliente do eXtreme Scale baseado em seus requisitos, incluindo a substituição de configurações.

É possível configurar um cliente doeXtreme Scale das seguintes maneiras:

- Configuração XML
- Configuração Programática
- Configuração da Estrutura Spring
- Desativando o Cache Local

É possível substituir os seguintes plug-ins em um cliente.

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

Configurando o Cliente com o XML

Um arquivo XML ObjectGrid pode ser usado para alterar configurações no lado do cliente. Para alterar as configurações em um cliente do eXtreme Scale, você deve criar um arquivo XML ObjectGrid que seja similar em estrutura ao arquivo que foi usado para o servidor do eXtreme Scale.

Suponha que o seguinte arquivo XML foi emparelhado com um arquivo XML de política de implementação, e esses arquivos foram usados para iniciar um servidor eXtreme Scale.

companyGridServerSide.xml

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

    <objectGrids>
        <objectGrid name="CompanyGrid">
            <bean id="TransactionCallback"
                className="com.company.MyTxCallback" />
            <bean id="ObjectGridEventListener"
                className="com.company.MyOgEventListener" />
            <backingMap name="Customer"
                pluginCollectionRef="customerPlugins"/>
            <backingMap name="Item" />
            <backingMap name="OrderLine" numberOfBuckets="1049"
                timeToLive="1600" ttlEvictorType="LAST_ACCESS_TIME" />
            <backingMap name="Order" lockStrategy="PESSIMISTIC"
                pluginCollectionRef="orderPlugins" />
        </objectGrid>
    </objectGrids>

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

Em um servidoreXtreme Scale, o CompanyGrid se comporta como definido pelo arquivo`companyGridServerSide.xml`. Por padrão, o cliente CompanyGrid tem as mesmas configurações do CompanyGrid que está executando no servidor. Entretanto, algumas das configurações podem ser substituídas no cliente.

Crie um ObjectGrid específico do cliente para substituir algumas dessas configurações. Copie o arquivo XML ObjectGrid que foi usado para iniciar o servidor e edite o arquivo para customizar o lado do cliente. Para remover um plug-in do cliente, use a cadeia vazia como o valor para o atributo className. Para

alterar um plug-in existente, especifique um novo valor para o atributo className. Um plug-in também pode ser incluído no arquivo desde que ele seja um dos plug-ins de substituição suportados.

Para definir um dos atributos no cliente, especifique um novo valor.

O seguinte arquivo XML ObjectGrid pode ser usado para especificar alguns dos atributos e plug-ins no cliente CompanyGrid.

companyGridClientSide.xml

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

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

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

O arquivo`companyGridClientSide.xml` substitui vários atributos e plug-ins no cliente CompanyGrid. No cliente, o TransactionCallback é com.company.MyClientTxCallback em oposição ao com.company.MyTxCallback, que executa no servidor. O ObjectGridEventListener é removido no cliente porque o valor de className é a cadeia vazia.

O backingMap do Cliente tem seu numberOfBuckets definido para 1429 no cliente. O Customer backingMap retém seu Evictor da configuração do servidor, mas o MapEventListener é removido do cliente.

O `numberOfBuckets` e `timeToLive` foram ajustados no lado do cliente do OrderLine backingMap.

O atributo lockStrategy neste arquivo é ignorado independentemente do valor especificado. Esse atributo não é suportado para substituição no lado do cliente.

Para criar o cliente CompanyGrid usando o arquivo `companyGridClientSide.xml`, passe o arquivo ObjectGrid XML como uma URL para um dos métodos de conexão no ObjectGridManager.

Criando o cliente para XML

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

Configurando o Cliente Programaticamente

Também é possível substituir as configurações do ObjectGrid do lado do cliente programaticamente. Crie um ObjectGridConfiguration que seja semelhante na estrutura para o ObjectGrid do lado do servidor. O seguinte código construirá um ObjectGrid do lado do cliente que é funcionalmente equivalente ao que seria criado usando o companyGridClientSide.xml fornecido na seção anterior.

Substituição do lado do cliente programaticamente

```
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);
```

Apenas os objetos ObjectGridConfiguration e BackingMapConfiguration que são incluídos no overrideMap serão verificados para ver se há plug-ins e atributos substituídos. Por exemplo, o código acima substituirá o número de depósitos no mapa OrderLine. Porém, o mapa Order permanecerá inalterado no lado do cliente porque nenhuma configuração para ele está incluída.

Configurando o Cliente com a Estrutura Spring

As configurações ObjectGrid do lado do cliente também podem ser substituídas usando a Estrutura Spring. O seguinte arquivo XML de exemplo mostra como criar um ObjectGridConfiguration e usá-lo para substituir alguma configuração do lado do cliente. Esse exemplo chama as mesmas APIs que são demonstradas na configuração Programática. O exemplo também é uma funcionalidade equivalente ao exemplo na configuração XML ObjectGrid.

Configuração do cliente com o 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>
```

Depois de criar o arquivo XML, carregue o arquivo e crie o ObjectGrid com o seguinte fragmento de código.

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

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

Consulte “Integração com a Estrutura Spring” na página 200 para obter informações adicionais.

Desativando o Cache Local

O cache local é ativado por padrão quando o bloqueio é configurado como otimista ou nenhum e não pode ser utilizado quando é configurado como pessimista.

Para desativar o cache local, você deve configurar o atributo numberOfBuckets como 0 no arquivo descritor do ObjectGrid de substituição do cliente.

Consulte as informações sobre bloqueio de entrada de mapa no *Guia de Administração* para obter mais informações.

Ativando o Mecanismo de Invalidação do Cliente

Em um ambiente do WebSphere eXtreme Scale distribuído, o lado do cliente possui um cache local por padrão ao utilizar a estratégia de bloqueio otimista ou quando o bloqueio está desativado. O cache perto tem seus próprios dados locais armazenados em cache. Se um eXtreme Scale cliente executar uma atualização, esta atualização chegará até seu cache perto e servidor do cliente. Entretanto, outros eXtreme Scale clientes não recebem as informações de atualização e poderão ter dados desatualizados.

Cache Local

Os aplicativos devem ficar cientes desse problema de dados obsoletos no cliente do eXtreme Scale. É possível usar a classe ObjectGridEventListener baseada em Java Message Service (JMS) integrado, com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener, para ativar o mecanismo de invalidação do cliente dentro de um ambiente do eXtreme Scale distribuído que é conhecido como uma grade eXtreme Scale.

O mecanismo de invalidação do cliente é a solução para o problema de dados antigos no cache local do cliente no ambiente do eXtreme Scale distribuído. Esse mecanismo garante que o cache local ao cliente esteja em sincronia com os servidores ou outros clientes. Contudo, mesmo com esse mecanismo de invalidação do cliente baseado em JMS, o cache local do cliente não é atualizado imediatamente. Um atraso ocorre quando o tempo de execução do eXtreme Scale publica atualizações.

Dois modelos estão disponíveis para o mecanismo de invalidação do cliente em um ambiente do eXtreme Scale distribuído:

- Modelo cliente-servidor: Neste modelo, todos os processos do servidor assumem a função de publicador e publicam todas as alterações na transação para o destino do JMS designado. Todos os processos do cliente estão nas funções de receptor e recebem todas as alterações na transação do destino do JMS designado.
- Modelo cliente como dupla função: Neste modelo, os processos do servidor não têm nada a fazer com o destino do JMS. Todos os processos do cliente assumem ambas as funções do JMS, publicador e receptor. As alterações na transação que ocorrem no cliente são publicadas no destino do JMS e todos os clientes recebem tais alterações.

Para obter mais informações sobre a ativação do mecanismo de invalidação do cliente, consulte “Listener de Eventos da JMS” na página 139.

Modelo de Cliente-servidor

Em um modelo cliente/servidor, os servidores exercem a função de publicador JMS e o cliente a função de receptor JMS.

Exemplo de XML de modelo cliente/servidor

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

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

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

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

</objectGridConfig>
```

Cliente como Modelo de Funções Duplas

No modelo cliente como dupla função, cada cliente assume ambas as funções do JMS, publicador e receptor. O cliente publica cada alteração da transação confirmada para um destino JMS designado e recebe todas as alterações transacionais confirmadas de outros clientes. O servidor não tem nada a fazer com o JMS neste modelo.

Exemplo de XML de modelo de função dupla

```
<?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>
```

Configuração de Tempo Limite de Nova Tentativa de Solicitação

Com mapas confiáveis, é possível fornecer um tempo limite de nova tentativa ao WebSphere eXtreme Scale. O tempo limite é usado junto com o tempo limite da transação para pedidos de transações de novas tentativas.

Há duas formas de configura mapas confiáveis. Se o valor for maior que zero, a solicitação é tentada até que a condição de tempo limite seja atendida ou uma falha permanente como uma exceção DuplicateKeyException seja encontrada. Se o valor for configurado para zero, o modo fail-fast será usado e o eXtreme Scale não tentará novamente.

Você fornece um valor de tempo limite em milissegundos no arquivo de propriedades do cliente ou na sessão. A sessão sempre substitui a definição das propriedades do cliente. Durante o tempo de execução, o tempo limite da transação é usado com o tempo limite de nova tentativa, garantindo que o tempo limite de nova tentativa não exceda o tempo limite da transação.

Devido às variações de como são concluídas as transações de auto-consolidação e sem auto-consolidação (transações que estão utilizando métodos begin e commit explícitos), as exceções válidas para a nova tentativa são diferentes.

Para transações que são chamadas dentro de uma sessão, a nova tentativa é válida para SystemExceptions do CORBA e exceções TargetNotAvailable do eXtreme Scale.

Para transações de autoconfirmação, a nova tentativa é válida para SystemExceptions do CORBA e Exceções de Disponibilidade do eXtreme Scale (ReplicationVotedToRollbackTransactionException, TargetNotAvailable, AvailabilityException e outras).

Para obter mais informações, consulte o tópico sobre o uso das sessões para acessar os dados na grade no *Guia de Programação*.

As falhas de aplicativo ou outras falhas permanentes são retornadas imediatamente e a transação não é tentada novamente. Essas falhas permanentes incluem as exceções DuplicateKeyException e KeyNotFoundException.

A configuração fail-fast retorna todas as exceções sem tentar novamente por nenhuma exceção.

A seguinte lista mostra as exceções em mais detalhes:

As exceções onde o cliente tenta novamente

- ReplicationVotedToRollbackTransactionException (somente na auto-consolidação)
- TargetNotAvailable
- org.omg.CORBA.SystemException
- AvailabilityException (somente na auto-consolidação)
- LockTimeoutException (somente na auto-consolidação)
- UnavailableServiceException (somente na auto-consolidação)

Outras exceções

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

Configuração da Propriedade requestRetryTimeout em um Arquivo de Propriedades

Para configurar o valor requestRetryTimeout em um cliente, inclua ou modifique a propriedade requestRetryTimeout no "Arquivo de Propriedades do Cliente" na página 195. As propriedades do cliente estão no arquivo `objectGridClient.properties` pelo padrão. A propriedade requestRetryTimeout é configurada em milissegundos. Configure o valor maior que zero para que o pedido seja tentado novamente em exceções para as quais uma nova tentativa está disponível. Configure o valor como `0` para falhar sem as novas tentativas em exceções. Para usar o comportamento padrão, remova a propriedade ou configure o valor como `-1`.

objectGridClient.properties

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

O valor requestRetryTimeout é especificado em milissegundos. No exemplo anterior, se o valor for usado em uma instância do ObjectGrid, o valor requestRetryTimeout será de 30.000 milissegundos ou de 30 segundos.

Configuração das Propriedades do Cliente Através da Obtenção de uma Conexão com ObjectGrid Programaticamente

Para configurar as propriedades do cliente programaticamente, crie primeiro um arquivo de propriedades do cliente. No exemplo a seguir, as propriedades do cliente estão no arquivo `objectGridClient.properties`. Depois de uma conexão ser estabelecida como ObjectGridManager, configure as propriedades do cliente. Em seguida, obtenha uma instância do ObjectGrid.As propriedades do cliente são configuradas nessa instância do ObjectGrid. Sempre que as propriedades do cliente forem alteradas, obtenha uma nova instância ObjectGrid.

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

Exemplo de Substituição de Sessão com Auto-consolidação

Para configurar o tempo limite de nova tentativa de solicitação em uma sessão ou para substituir a propriedade do cliente requestRetryTimeout, chame o método setRequestRetryTimeout(long) na interface Session.

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

Esta sessão agora usa um valor requestRetryTimeout de 30.000 milissegundos ou de 30 segundos, independente do valor configurado no arquivo de propriedades do cliente. Para obter mais informações sobre a interface de sessão, consulte Usando sessões para acessar dados na grade.

Resolução de Problemas de Configuração XML

Ocasionalmente, ao configurar o eXtreme Scale, é possível encontrar o comportamento que não é esperado.

Na seção a seguir são apresentados diversos problemas de configuração XML que podem ocorrer.

Política de Implementação e Arquivos XML do Incompatíveis

A política de implementação e os arquivos XML de ObjectGrid devem corresponder. Se eles não possuem nomes de ObjectGrid e nomes de mapas correspondentes, ocorrem erros.

BackingMap e referências de mapa incorretos

Se a lista de backingMap em um arquivo XML de ObjectGrid não corresponder à lista de referências de mapas em um arquivo XML da política de implementação, ocorre um erro no servidor de catálogos.

Por exemplo, o seguinte arquivo XML de ObjectGrid e arquivo XML da política de implementação são utilizados para iniciar um processo de contêiner. O arquivo da política de implementação possui mais referências de mapas que são listados no arquivo XML de ObjectGrid.

ObjectGrid.xml - exemplo incorreto

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

deploymentPolicy.xml - exemplo incorreto

```
<?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>
```

Mensagens

Ocorre uma ObjectGridException causada por uma exceção de IncompatibleDeploymentPolicyException. Para o exemplo precedente, a exceção é exibida como a mensagem a seguir:

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.

Se a política de implementação não possuir as referências de mapas para os backingMaps listados no arquivo XML de ObjectGrid, um ObjectGridException também ocorre com uma exceção de IncompatibleDeploymentPolicyException. Por exemplo:

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.

Problema

A lista backingMap no arquivo XML de ObjectGrid e as referências de mapas na política de implementação devem ser correspondentes.

Solução

Determine qual lista é correta para seu ambiente e corrija o arquivo XML que contém a lista incorreta.

Nomes de ObjectGrid incorretos

O nome do ObjectGrid é referido no arquivo XML ObjectGrid e no arquivo XML da política de implementação.

Mensagem

Ocorre uma ObjectGridException causada por uma exceção de IncompatibleDeploymentPolicyException. A seguir, está um exemplo.

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

Problema

O arquivo XML ObjectGrid é a lista principal dos nomes ObjectGrid. Ocorrerá um erro se uma política de implementação possuir um nome ObjectGrid que não está contido no arquivo XML ObjectGrid.

Solução

Verifique se o nome ObjectGrid está correto. Remova quaisquer nomes extras ou inclua nomes ObjectGrid ausentes nos arquivos XML ObjectGrid ou da política de implementação. Na mensagem de exemplo, o objectGridName está escrito incorretamente como ″accountin″ em vez de ″accounting″.

O Valor de Atributo XML não é Válido

A alguns dos atributos no arquivo XML podem ser designados apenas alguns valores. Estes atributos possuem valores aceitáveis enumerados pelo esquema. A lista a seguir fornece alguns dos atributos:

- atributo authorizationMechanism no elemento objectGrid
- atributo copyMode no elemento backingMap
- atributo lockStrategy no elemento backingMap
- atributo ttlEvictorType no elemento backingMap
- atributo type no elemento property
- initialState no elemento objectGrid
- evictionTriggers no elemento backingMap

Se a um destes atributos for designado um valor inválido, a validação XML falhará. No arquivo XML de exemplo a seguir, é utilizado um valor incorreto de INVALID_COPY_MODE:

Exemplo de INVALID_COPY_MODE

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

A mensagem a seguir aparece no log.

CWOBJ2403E: O arquivo XML é inválido. Foi detectado um problema com < null > na linha 5. A mensagem de erro é cvc-enumeration-valid: O valor 'INVALID_COPY_MODE' não possui uma faceta válida com relação à enumeração'[COPY_ON_READ_AND_COMMIT, COPY_ON_READ, COPY_ON_WRITE, NO_COPY,COPY_TO_BYTES]'. Ela deve ser um valor a partir da enumeração.

Atributos ou Tags Ausentes

Ocorrem erros se um arquivo XML não possuir os atributos ou tags corretos. Por exemplo, o seguinte arquivo XML do ObjectGrid não possui a tag final < /objectGrid >:

atributos ausentes - XML de exemplo

```
<?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>
```

A mensagem a seguir aparece no log.

CWOBJ2403E: O arquivo XML é inválido. Foi detectado um problema com < null > na linha 7. A mensagem de erro é: The end-tag for element type ″objectGrid″ must end with a '>' delimiter.

Ocorre uma ObjectGridException sobre o arquivo XMl inválido com o nome do arquivo XML

Erros de Sintaxe

Se um arquivo XML é formatado com sintaxe incorreta ou ausente, o CWOBJ2403E aparece no log. Por exemplo, a mensagem a seguir é exibida quando um sinal de aspas está ausente em um dos atributos XML.

CWOBJ2403E: O arquivo XML é inválido. Foi detectado um problema com < null > na linha 7. A mensagem de erro é: Open quote is expected for attribute ″maxSyncReplicas″ associated with an element type ″mapSet″.

An ObjectGridException about the invalid XML file also occurs.

Fazendo Referência a uma Coleta de Plug-in Inexistente

Ao utilizar o XML para definir plug-ins do BackingMap, o atributo pluginCollectionRef do elemento backingMap deve fazer referência a um backingMapPluginCollection. O atributo pluginCollectionRef deve corresponder com o ID de um dos elementos backingMapPluginCollection.

Mensagem

Se o atributo pluginCollectionRef não corresponder a nenhum dos atributos de ID de algum dos elementos backingMapPluginConfiguration, uma mensagem semelhante à seguinte será exibida no log.

```
[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'.
```

Problema

O arquivo XML a seguir é utilizado para produzir o erro. Observe que o nome do BackingMap book possui seu atributo pluginCollectionRef configurado como bookPlugins e o backingMapPluginCollection único possui um ID de collection1.

Referenciando um XML de atributo não-existente - exemplo

```
<?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>
```

Solução

Para corrigir o problema, certifique-se de que o valor de cada pluginCollectionRef corresponda ao ID de um dos elementos backingMapPluginCollection. Simplesmente altere o nome de pluginCollectionRef para collection1 para não receber este erro. Outras maneiras de corrigir o problema incluem a alteração do ID do backingMapPluginCollection existente para corresponder ao pluginCollectionRef ou a inclusão de um backingMapPluginCollection adicional com um ID que corresponda ao pluginCollectionRef.

Validando XML sem Suporte de uma Implementação

O IBM Software Development Kit (SDK) Versão 1.4.2 contém uma implementação de alguma função JAXP (Java API for XML Processing) para usar para validação XML contra um esquema.

Ao utilizar um SDK que não contém esta implementação, as tentativas de validação poderão falhar. Se desejar validar o XML usando um SDK que não contém esta implementação, efetue download do Apache Xerces e inclua seus arquivos JAR (Java archive) no caminho de classe.

Ao tentar validar o XML com um SDK que não possui a implementação necessária, o log contém o seguinte erro:

```
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)...
```

O SDK utilizado não contém uma implementação da função JAXP necessária para validar arquivos XML em um esquema.

Para evitar este problema, depois de efetuar download do Xerces e incluir os arquivos JAR no caminho de classe,é possível validar o arquivo XML com êxito.

Utilitários de Carga

Com um plug-in de utilitário de carga do eXtreme Scale, um mapa do eXtreme Scale poderá se comportar como um cache de memória para dados que são tipicamente mantidos em um armazenamento persistente no mesmo sistema ou em algum outro sistema. Geralmente, um banco de dados ou sistema de arquivos é utilizado como o armazenamento persistente. Uma JVM (Java Virtual Machine) também pode ser usada como a origem de dados, permitindo que caches baseados em hub seja construído usando o eXtreme Scale. Um utilitário de carga possui a lógica para leitura e escrita de dados no armazenamento persistente.

Visão Geral

Os utilitários de carga são plug-ins de mapa de apoio que são chamados quando são feitas alterações no mapa de apoio ou quando o mapa de apoio não pode atender a um pedido de dados (um erro de cache). Consulte as informações sobre cenários de armazenamento em cache no *Visão Geral do Produto* para obter uma visão geral sobre como o eXtreme Scale pode interagir com um utilitário de carga.

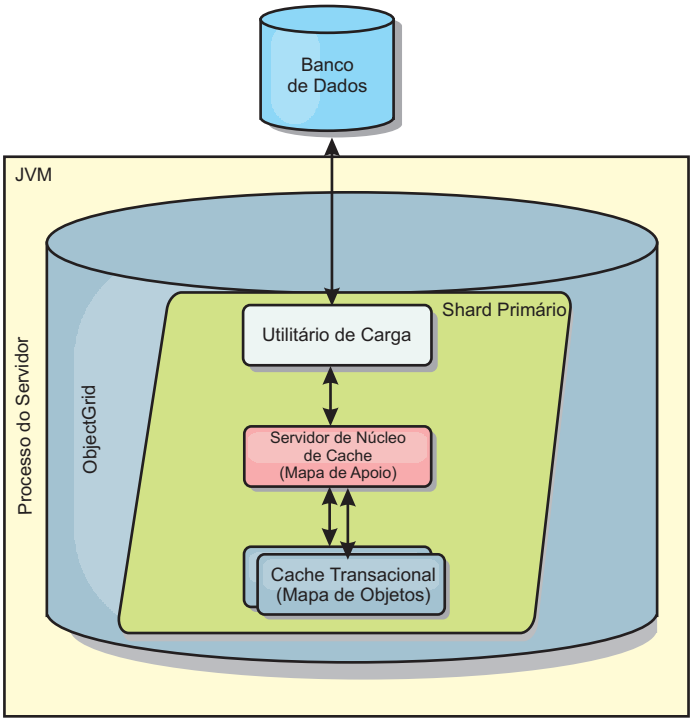


Figura 1. Utilitário de Carga

Dois utilitários de carga integrados podem simplificar muito a integração com back ends de banco de dados relacional. Os utilitários de carga JPA utilizam os recursos ORM (Object-Relational Mapping) de ambas as implementações OpenJPA e Hibernate da especificação JPA (Java Persistence API). Consulte as informações sobre utilitários de carga JPA no *Visão Geral do Produto* para obter informações adicionais.

Utilizando um Utilitário de Carga

Para incluir um Utilitário de Carga na configuração do BackingMap, é possível utilizar a configuração programática ou a configuração do XML. Um utilitário de carga possui o seguinte relacionamento com um mapa de apoio.

- Um mapa de apoio pode ter apenas um utilitário de carga.
- Um mapa de apoio de cliente (cache local) não pode ter um utilitário de carga.
- Um definição do utilitário de carga pode ser aplicada a múltiplos mapas de apoios, mas cada mapa de apoio possui sua própria instância do utilitário de carga.

Para obter mais informações, consulte o tópico sobre gravação de um utilitário de carga no *Visão Geral do Produto*.

Abordagem da Configuração XML para Conectar um Utilitário de Carga

Um utilitário de carga fornecido pelo aplicativo pode ser plugado usando um arquivo XML. O exemplo a seguir demonstra como conectar o utilitário de carga ″MyLoader″ ao mapa de apoio ″map1″: É necessário especificar o className para seu utilitário de carga, os detalhes de nome e conexão do banco de dados e as propriedades do nível de isolamento. É possível usar a mesma estrutura XML se você estiver usando apenas um pré-utilitário de carga ao especificar o nome da classe de pré-utilitário de carga em vez de um nome de classe de utilitário de carga completo.

Configuração do utilitário de carga comXML
```
<?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>
```

Conectando um Utilitário de Carga Programaticamente

É possível usar somente uma configuração programática com grades locais na memória. O trecho de código a seguir demonstra como conectar o Utilitário de Carga fornecido pelo aplicativo ao mapa de apoio para map1, por meio da API do ObjectGrid:

Configuração programática de um Utilitário de Carga
```
import com.ibm.websphere.objectgrid.ObjectGridManagerFactory;
import com.ibm.websphere.objectgrid.ObjectGridManager;
import com.ibm.websphere.objectgrid.ObjectGrid;
import com.ibm.websphere.objectgrid.BackingMap;
ObjectGridManager ogManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid og = ogManager.createObjectGrid("grid");
BackingMap bm = og.defineMap( "map1" );
```

```
MyLoader loader = new MyLoader();
loader.setDataBaseName("testdb");
loader.setIsolationLevel("read committed");
bm.setLoader( loader );
```

Este fragmento assume que a classe MyLoader é a classe fornecida pelo aplicativo que implementa a interface com.ibm.websphere.objectgrid.plugins.Loader. Como a associação de um Loader com um mapa de suporte não pode ser alterada após a inicialização de um ObjectGrid, o código deve ser executado antes de chamar o método initialize da interface ObjectGrid que está sendo chamada. Uma exceção IllegalStateException ocorre em uma chamada de método setLoader se ela for chamada depois de a inicialização ocorrer.

O Utilitário de Carga fornecido pelo aplicativo pode ter propriedades configuradas. No exemplo, o utilitário de carga MyLoader é utilizado para ler e gravar dados de uma tabela em um banco de dados relacional. O utilitário de carga deve especificar o nome do banco de dados e o nível de isolamento do SQL. O loader MyLoader possui os métodos setDataBaseName e setIsolationLevel que permitem que o aplicativo configure estas duas propriedades do Loader.

Considerações sobre o Loader

Este tópico descreve considerações ao implementar um utilitário de carga.

Considerações sobre Pré-carregamento

Os utilitários de carga são plug-ins de mapa de apoio que são chamados quando são feitas alterações no mapa de apoio ou quando o mapa de apoio não pode atender a um pedido de dados (um erro de cache). Para obter uma visão geral de como o eXtreme Scale interage com um utilitário de carga, consulte as informações sobre cenários de armazenamento em cache sequencial no *Visão Geral do Produto*

Cada mapa de suporte possui um atributo booleano preloadMode que pode ser configurado para indicar se o pré-carregamento de um mapa foi concluído de modo assíncrono. Por padrão, o atributo preloadMode está configurado como false, o que indica que a inicialização do mapa de suporte não será concluída até que o pré-carregamento do mapa esteja concluído. Por exemplo, a inicialização do mapa de suporte não será concluída até que o método preloadMap seja retornado. Se o método preloadMap ler uma grande quantidade de dados no seu back end e carregá-los para o mapa, o tempo de conclusão desse procedimento pode ser relativamente longo. Neste caso, é possível configurar um mapa de suporte para utilizar o pré-carregamento assíncrono do mapa, configurando o atributo preloadMode como true. Essa configuração faz o código de inicialização do mapa de suporte efetuar spawn de um encadeamento que chama o método preloadMap, permitindo a conclusão da inicialização de um mapa de suporte enquanto o pré-carregamento do mapa ainda está em andamento.

O trecho de código a seguir ilustra como o atributo preloadMode é configurado para ativar o pré-carregamento assíncrono:

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

O atributo preloadMode também pode ser configurado utilizando um arquivo XML conforme ilustrado no seguinte exemplo:

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

TxID e Utilização da Interface TransactionCallback

O método get e os métodos batchUpdate da interface do Utilitário de Carga são transmitidos para um objeto TxID que representa a transação do objeto Session que requer que a operação get ou batchUpdate seja executada. É possível que os métodos get e batchUpdate sejam chamados mais de uma vez por transação. Portanto, os objetos com escopo definido pela transação requeridos pelo Loader geralmente são mantidos em um slot do objeto TxID. Um utilitário de carga JDBC (Java Database Connectivity) é usado para ilustrar como um utilitário de carga usa as interfaces TxID e TransactionCallback.

Também é possível que vários mapas do ObjectGrid sejam armazenados no mesmo banco de dados. Cada mapa possui seu próprio Loader e cada Loader pode precisar conectar-se ao mesmo banco de dados. Ao conectar-se ao mesmo banco de dados, cada Loader deseja utilizar a mesma conexão JDBC para que as alterações em cada tabela sejam confirmadas como parte da transação do mesmo banco de dados. Geralmente, a mesma pessoa que grava a implementação do Loader também grava a implementação do TransactionCallback. O melhor método é quando a interface TransactionCallback é estendida de modo a incluir os métodos que Utilitário de Carga precisa para obter uma conexão com o banco de dados e para armazenar em cache as instruções preparadas. O motivo para esta metodologia torna-se aparente à medida que você visualiza como as interfaces TransactionCallback e TxID são utilizadas pelo utilitário de carga.

Como um exemplo, o utilitário de carga pode precisar da interface TransactionCallback para ser estendido conforme a seguir:

```
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 );
}
```

Com tais novos métodos, os métodos get e batchUpdate do Utilitário de Carga podem obter uma conexão da seguinte forma:

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

No exemplo anterior e nos exemplos que seguem, vTcb e ivOcb são variáveis da instância do Carregador que foram inicializadas conforme descrito na seção Considerações de Pré-carregamento. A variável ivTcb é uma referência à instância MyTransactionCallback e ivOcb é uma referência à instância MyOptimisticCallback. A variável databaseName é uma variável da instância do Utilitário de Carga que foi configurada como uma propriedade do Utilitário de Carga durante a inicialização do mapa de suporte. O argumento isolationLevel é uma das constantes da Conexão JDBC que estão definidas para os diversos níveis de isolamento suportados pelo JDBC. Se o Utilitário de Carga estiver utilizando uma implementação otimista, o método get geralmente utilizará uma conexão de

autoconfirmação JDBC para buscar os dados do banco de dados. Nesse caso, o Utilitário de Carga pode ter um método getAutoCommitConnection que seja implementado da seguinte forma:

```
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;
}
```

Lembre-se de que o método batchUpdate possui a seguinte instrução switch:

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

Cada um dos métodos buildBatchSQL utiliza a interface MyTransactionCallback para obter uma instrução preparada. Este é um trecho de código que mostra o método buildBatchSQLUpdate construindo uma instrução SQL update para atualizar uma entrada EmployeeRecord e incluindo-a na atualização de batch:

```
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();
}
```

Quando o loop batchUpdate tiver construído todas as instruções preparadas, ele chamará o método getPreparedStatementCollection. Esse método é implementado como segue:

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

Quando o aplicativo chama o método commit no objeto Session, o código do Session chamará o método commit, no método TransactionCallback, depois de enviar todas as alterações feitas pela transação fora do Utilitário de Carga para cada mapa alterado pela transação. Como todos os Loaders utilizaram o método MyTransactionCallback para obter todas as conexões e instruções preparadas necessárias, o método TransactionCallback sabe qual conexão utilizar para solicitar que o back end confirme as alterações. Portanto, estender a interface TransactionCallback com métodos requeridos por cada um dos Loaders tem as seguintes vantagens:

- O objeto TransactionCallback encapsula a utilização de slots TxID para dados com escopo definido pela transação e o Loader não requer informações sobre os slots TxID. O Loader apenas precisa saber sobre os métodos que foram incluídos

no TransactionCallback utilizando a interface MyTransactionCallback para as funções de suporte requeridas pelo Loader.

- O objeto TransactionCallback pode assegurar que o compartilhamento da conexão ocorra entre cada Loader que se conecta ao mesmo backend para que um protocolo de confirmação de duas fases seja evitado.
- O objeto TransactionCallback pode assegurar que a conexão com o backend seja orientada para conclusão por meio de uma confirmação ou rollback chamado na conexão quando apropriado.
- O TransactionCallback garante a limpeza dos recursos do banco de dados após a conclusão de uma transação.
- O TransactionCallback fica oculto se ele estiver obtendo uma conexão gerenciada a partir de um ambiente gerenciado como WebSphere Application Server ou algum outro servidor de aplicativos compatível com Java 2 Platform, Enterprise Edition (J2EE). Esta vantagem permite que o mesmo código do Loader seja utilizado em ambientes gerenciados e não gerenciados. Apenas o plug-in TransactionCallback deve ser alterado.
- Para obter informações detalhadas sobre como a implementação do TransactionCallback utiliza os slots TxID para dados dentro do escopo da transação, consulte Plug-in do TransactionCallback.

OptimisticCallback

Conforme mencionado anteriormente, o Utilitário de Carga pode utilizar uma abordagem otimista para controle de simultaneidade. Nesse caso, o exemplo do método buildBatchSQLUpdate precisará ser modificado ligeiramente para implementar uma abordagem otimista. Existem várias maneiras possíveis para utilizar uma abordagem otimista. Uma maneira típica é ter uma coluna de time stamp ou uma coluna do contador de números de seqüência para o controle de versões de cada atualização da linha. Suponha que a tabela de funcionários tenha uma coluna de números de seqüência que aumenta sempre que a linha é atualizada. Em seguida, você modifica a assinatura do método buildBatchSQLUpdate para que ela seja transmitida para o objeto LogElement em vez do par chave e valor. Ele também precisa utilizar o objeto OptimisticCallback que está conectado ao mapa de suporte para obter o objeto da versão inicial e para atualizar o objeto da versão. Este é um exemplo de método buildBatchSQLUpdate modificado que utiliza a variável da instância ivOcb inicializada conforme descrito na seção preloadMap:

Exemplo de código de método de atualização de lote modificado

```
private void buildBatchSQLUpdate( TxID tx, LogElement le, Connection conn )
 throws SQLException, LoaderException
{
    // Obter o objeto da versão inicial quando esta entrada do mapa foi lida pela última vez ou
    // atualizada no banco de dados.
    Employee emp = (Employee) le.getCurrentValue();
    long initialVersion = ((Long) le.getVersionedValue()).longValue();
    // Obter o objeto da versão de Employee atualizado para a operação SQL
    //update.
    Long currentVersion = (Long)ivOcb.getVersionedObjectForValue( emp );
    long nextVersion = currentVersion.longValue();
    // Agora construa o SQL update que inclui o objeto de versão na cláusula where
    // para verificação otimista.
    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();
}
```

O exemplo mostra que o LogElement é utilizado para obter o valor de versão inicial. Quando a transação acessa pela primeira vez a entrada do mapa, é criado um LogElement com o objeto Employee inicial obtido do mapa. O objeto Employee inicial também é transmitido para o método getVersionedObjectForValue na interface OptimisticCallback e o resultado é salvo no LogElement. Este processamento ocorre antes de um aplicativo receber uma referência ao objeto Employee inicial e de chamar algum método que altere o estado do objeto Employee inicial.

O exemplo mostra que o Loader utiliza o método getVersiondObjectForValue para obter o objeto de versão para o objeto Employee atual atualizado. Antes de chamar o método batchUpdate na interface do utilitário de carga, eXtreme Scale chama o método updateVersionedObjectForValue na interface OptimisticCallback para gerar um novo objeto de versão a ser gerado para o objeto Employee atualizado. Quando o método batchUpdate retornar ao ObjectGrid, o LogElement será atualizado com o objeto de versão atual e se tornará o novo objeto de versão inicial. Esta etapa é necessária porque o aplicativo pode ter chamado o método flush no mapa em vez do método commit na Session. É possível que o Loader seja chamado várias vezes por uma única transação para a mesma chave. Por este motivo, o eXtreme Scale assegura que o LogElement seja atualizado com o novo objeto de versão toda vez que a linha for atualizada na tabela de funcionários.

Agora que o Utilitário de Carga tem o objeto de versão inicial e o próximo objeto de versão, ele pode executar uma instrução SQL update que configura a coluna SEQNO para o próximo valor do objeto de versão e utiliza o valor do objeto de versão inicial na cláusula where. Essa abordagem, às vezes, é referida como uma instrução update super qualificada. A utilização da instrução update super qualificada permite que o banco de dados relacional verifique se a linha não foi alterada por alguma outra transação entre o tempo de leitura do banco de dados por parte da transação e o momento em que esta o atualizou. Se outra transação modificou a linha, a matriz de contagem retornada pela atualização de batch indica que zero linhas foram atualizadas para esta chave. O Utilitário de Carga é responsável por verificar se a operação SQL update atualizou a linha. Se isso não ocorreu, ele exibirá uma exceção com.ibm.websphere.objectgrid.plugins.OptimisticCollisionException para informar o objeto Session que o método batchUpdate falhou porque mais de uma transação simultânea está tentando atualizar a mesma linha na tabela de banco de dados. Esta exceção faz a Session efetuar rollback e o aplicativo deve tentar novamente a transação inteira. O fundamento lógico é que a nova tentativa será bem-sucedida, motivo pelo qual esta abordagem é chamada de otimista. A abordagem otimista apresenta um desempenho melhor quando os dados não sofrem alterações freqüentes ou transações simultâneas raramente tentam atualizar a mesma linha.

É importante que o Utilitário de Carga utilize o parâmetro key do construtor OptimisticCollisionException para identificar qual chave ou conjunto de chaves causou a falha do método batchUpdate otimista. O parâmetro de chave pode ser o próprio objeto de chave ou uma matriz de objetos de chave se mais de uma chave resultar em uma falha de atualização otimista. E o eXtreme Scale usa o método getKey do construtor OptimisticCollisionException para determinar quais entradas de mapa contêm dados desatualizados e causaram a exceção. Parte do processamento de rollback é liberar cada entrada do mapa stale do mapa. A liberação de entradas stale é necessária para que qualquer transação subseqüente que acessa a mesma chave ou chaves resulte no método get da interface do Loader que está sendo chamada para atualizar as entradas do mapa com os dados atuais do banco de dados.

Outras maneiras para um Loader implementar uma abordagem otimista incluem:

- Não existe nenhuma coluna de time stamp ou de número de seqüência. Neste caso, o método getVersionObjectForValue na interface OptimisticCallback apenas retorna o próprio objeto de valor como a versão. Com essa abordagem, o Utilitário de Carga precisa construir uma cláusula WHERE que inclua cada um dos campos do objeto de versão inicial. Essa abordagem não é eficiente e nem todos os tipos de colunas estão qualificados para serem utilizados na cláusula WHERE de uma instrução SQL update super qualificada. Esta abordagem geralmente não é utilizada.
- Não existe nenhuma coluna de time stamp ou de número de seqüência. No entanto, diferente da abordagem anterior, a cláusula WHERE contém apenas os campos de valores modificados pela transação. Um método para detectar quais campos foram modificados é configurar o modo de cópia no mapa de suporte como CopyMode.COPY_ON_WRITE. Esse modo de cópia requer que uma interface de valor seja transmitida para o método setCopyMode na interface BackingMap. O BackingMap cria objetos de proxy dinâmicos que implementam a interface de valor fornecida. Com este modo de cópia, o Loader pode lançar cada valor em um objeto com.ibm.websphere.objectgrid.plugins.ValueProxyInfo. A interface ValueProxyInfo possui um método que permite que o Loader obtenha a Lista de nomes de atributos que foram alterados pela transação. Esse método permite que o Utilitário de Carga chame os métodos get na interface de valor para os nomes de atributos a fim de obter os dados alterados e construa uma instrução SQL update que configure apenas os atributos alterados. A cláusula WHERE agora pode ser construída de modo a ter a coluna-chave primária e cada uma das colunas de atributos alteradas. Esta abordagem é mais eficiente do que a abordagem anterior, mas requer que mais código seja gravado no Loader e gera a possibilidade de que o cache de instrução preparado precise ser maior para manipular as diferentes permutações. No entanto, se as transações geralmente modificarem apenas alguns dos atributos, esta limitação pode não ser um problema.
- Alguns bancos de dados relacionais podem ter uma API para ajudar na manutenção automática de dados da coluna que são úteis para o controle de versões otimista. Consulte a documentação do banco de dados para determinar se existe esta possibilidade.

Configurando Utilitários de Carga do JPA

Um Utilitário de Carga do Java Persistence API (JPA) é uma implementação do plug-in do utilitário de carga que usa o JPA para interagir com o banco de dados. Para configurar um utilitário de carga, é necessário configurar os parâmetros necessários e fazer atualizações nos arquivos de configuração XML.

Antes de Iniciar

- É necessário ter uma implementação do JPA, como Hibernate ou OpenJPA.
- O banco de dados pode ser qualquer back end que seja suportado pelo provedor JPA escolhido.
- É possível usar o plug-in do JPALoader ao armazenar dados usando a API ObjectMap. Use o plug-in do JPAEntityLoader ao armazenar dados usando a API EntityManager.

Por Que e Quando Desempenhar Esta Tarefa

Para obter mais informações sobre como o Java Persistence API (JPA) Loader funciona, consulte as informações no *Visão Geral do Produto*.

1. Configure os parâmetros necessários que o JPA requer para interagir com o banco de dados.

   Os seguintes parâmetros são necessários. Esses parâmetros são configurados no bean JPALoader ou JPAEntityLoader e no bean JPATxCallback.

   - **persistenceUnitName**: Especifica o nome da unidade de persistência. Esse parâmetro é necessário para duas finalidades: criar um EntityManagerFactory do JPA e localizar os metadados da entidade JPA no arquivo `persistence.xml`. Este atributo é configurado no bean JPATxCallback.
   - **JPAPropertyFactory**: Especifica o factory para criar um mapa de propriedade de persistência para substituir as propriedades de persistência padrão. Este atributo é configurado no bean JPATxCallback. Para configurar esse atributo, a configuração de estilo Spring é necessária.
   - **entityClassName**: Especifica o nome da classe de entidade que é necessário para usar os métodos JPA, como EntityManager.persist, EntityManager.find e assim por diante. O JPALoader requer este parâmetro, mas o parâmetro é opcional para **JPAEntityLoader**. No caso do JPAEntityLoader, se um parâmetro **entityClassName** não estiver configurado, a classe de entidade configurada no mapa de entidade ObjectGrid será usada. A mesma classe deve ser usada para o eXtreme Scale EntityManager e para o provedor JPA. Este atributo é configurado no bean JPALoader ou JPAEntityLoader.
   - **preloadPartition**: Especifica a partição na qual o pré-carregamento do mapa será iniciado. Se a partição do pré-carregamento for menor que zero ou maior que o número total de partições menos 1, o pré-carregamento do mapa não será iniciado. O valor padrão é -1, significando que o pré-carregamento não é iniciado por padrão. Este atributo é configurado no bean JPALoader ou JPAEntityLoader.

   Com exceção dos quatro parâmetros JPA a serem configurados no eXtreme Scale, os metadados JPA são utilizados para recuperar a chave das entidades JPA. Os metadados JPA podem ser configurados como anotação ou como um arquivo `orm.xml` especificado no arquivo `persistence.xml`. Eles não fazem parte da configuração do eXtreme Scale.

2. Configure os arquivos XML para a configuração do JPA.

   Para configurar o JPALoader ou o JPAEntityLoader, consulte as informações sobre os plug-ins do utilitário de carga no *Guia de Programação*.

   Configure um retorno de chamada de transação JPATxCallback junto com a configuração do utilitário de carga. O seguinte exemplo mostra um arquivo descritor XML ObjectGrid (`objectgrid.xml`), que tem o JPAEntityLoader e o JPATxCallback configurados:

Configurando um utilitário de carga incluindo um retorno de chamada - XML de exemplo

```
<?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>
```

Se desejar configurar um JPAPropertyFactory, será necessário usar uma configuração de estilo Spring. A seguir há uma amostra de arquivo de configuração XML, `JPAEM_spring.xml`, que configura um bean Spring a ser usado para as configurações do eXtreme Scale.

Configurando um utilitário de carga incluindo um factory de propriedade JPA - XML de exemplo

```
<?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>
```

O arquivo XML de configuração `Objectgrid.xml` vem a seguir. Observe que o nome do ObjectGrid é JPAEM, que corresponde ao nome do ObjectGrid no arquivo de configuração Spring `JPAEM_spring.xml`.

Configuração de um utilitário de carga JPAEM - XML de exemplo

```
<?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>
```

Uma entidade pode ser anotada com as anotações do JPA e com as anotações do gerenciador de entidade do eXtreme Scale. Cada anotação possui um XML equivalente que pode ser utilizado. Assim, o eXtreme Scale incluiu o espaço de nomes Spring. Também é possível configurá-los utilizando o suporte a espaço de nomes Spring. Para obter mais informações sobre o suporte ao espaço de nomes Spring, consulte “Integração com a Estrutura Spring” na página 200.

Configurando um Atualizador de Dados Baseado em Tempo do JPA

É possível configurar uma atualização de banco de dados baseado em tempo usando o XML para uma configuração do eXtreme Scale local ou distribuída. Também é possível definir uma configuração local programaticamente.

Por Que e Quando Desempenhar Esta Tarefa

Para obter mais informações sobre como o atualizador de dados baseado em tempo do Java Persistence API (JPA) trabalha, consulte as informações no *Guia de Programação*.

Criar uma configuração timeBasedDBUpdate.

- **Com o arquivo XML:**

  O seguinte exemplo mostra um arquivo `objectgrid.xml` que contém uma configuração timeBasedDBUpdate:

```
Atualizador baseado em tempo do JPA - exemplo de XML
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
        <objectGrid name="changeOG"
            entityMetadataXMLFile="userEMD.xml">
            <backingMap name="user" >
                <timeBasedDBUpdate timestampField="rowChgTs"
                      persistenceUnitName="userderby"
                      entityClass="com.test.UserClass"
                      mode="INVALIDATE_ONLY"
                />
            </backingMap>
        </objectGrid>
    </objectGrids>
    <backingMapPluginCollections>
</objectGridConfig>
```

  Neste exemplo, o mapa ″user″ é configurado com a atualização do banco de dados baseada em tempo. O modo de atualização de banco de dados é INVALIDATE_ONLY e o campo do registro de data e hora é rowChgTs.

  Quando um ObjectGrid ″changeOG″ distribuído é iniciado no servidor de contêiner, um encadeamento de atualização de banco de dados baseado em tempo é iniciado automaticamente na partição 0.

- **Programaticamente:**

  Se você criar um ObjectGrid local, também é possível criar um objeto TimeBasedDBUpdateConfig e configurá-lo na instância BackingMap:

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

  Para obter mais informações sobre como configurar um objeto na instância BackingMap, consulte as informações sobre a interface BackingMap na Documentação de API.

  Por outro lado, não é possível anotar o campo do registro de data e hora na classe de entidade usando a anotação com.ibm.websphere.objectgrid.jpa.dbupdate.annotation.Timestamp. Ao configurar o valor na classe, não é necessário configurar o timestampField na configuração XML.

O que Fazer Depois

Inicie um atualizador de dados baseado em tempo do JPA. Consulte as informações sobre a iniciação de um atualizador de dados baseado em tempo do JPA no *Guia de Programação* para obter mais informações.

Plug-in do Cache JPA

O WebSphere eXtreme Scale inclui plug-ins de cache de nível 2 (L2) para ambos os provedores OpenJPA e Hibernate Java Persistence API (JPA).

Usar o eXtreme Scale como um provedor de cache L2 aumenta o desempenho na leitura e consulta de dados e reduz a carga para o banco de dados. O WebSphere eXtreme Scale tem vantagens sobre as implementações de cache integradas porque o cache é automaticamente replicado entre todos os processos. Quando um cliente armazena em cache um valor, todos os outros clientes conseguem usar o valor armazenado em cache que está localmente na memória.

Com os plug-ins de cache do ObjectGrid OpenJPA e Hibernate, é possível criar três tipos de topologia: integrada, integrada-particionada e remota.

Topologia Integrada

Uma topologia integrada cria um servidor eXtreme Scale dentro do espaço do processo de cada aplicativo. O OpenJPA e o Hibernate lêem a cópia na memória do cache diretamente e gravam para todas as outras cópias. É possível melhorar o desempenho de gravação usando replicação assíncrona. Esta topologia padrão executa melhor quando a quantidade de dados armazenados em cache é pequena o suficiente para ajustar-se a um único processo.

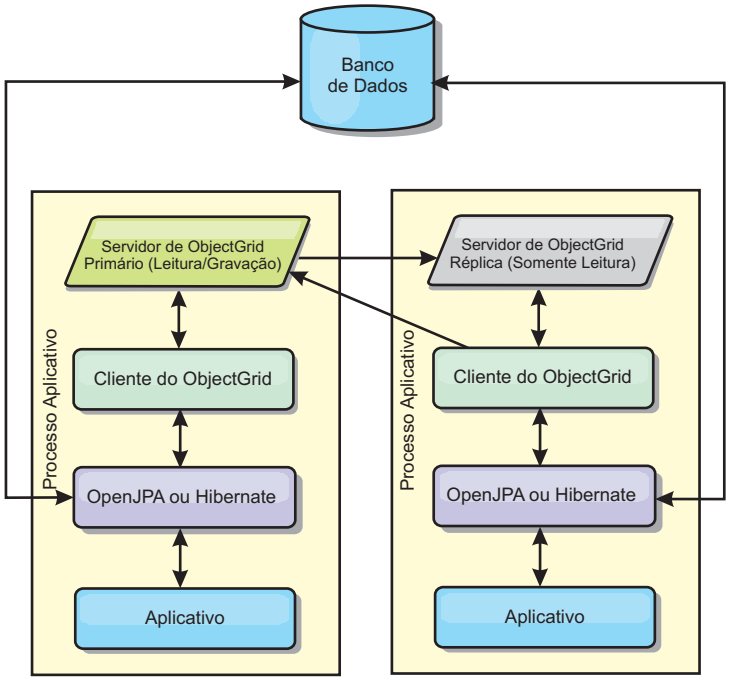


Figura 2. Topologia Integrado do JPA

Vantagens:

- Todas as leituras de cache são muito rápidas, com acessos locais.
- Simples para configurar.

Limitações:

- A quantidade de dados é limitada ao tamanho do processo.
- Todas as atualizações de cache são enviadas para um processo.

Topologia Integrada e Particionada

Quando os dados armazenados em cache são muito grandes para ajustar-se em um único processo, a topologia integrada e particionada utiliza partições do ObjectGrid

para dividir os dados sobre vários processos. O desempenho não é tão alto quanto o da topologia integrada porque a maioria dos caches é remota. Porém, ainda é possível usar esta opção quando a latência do banco de dados é alta.

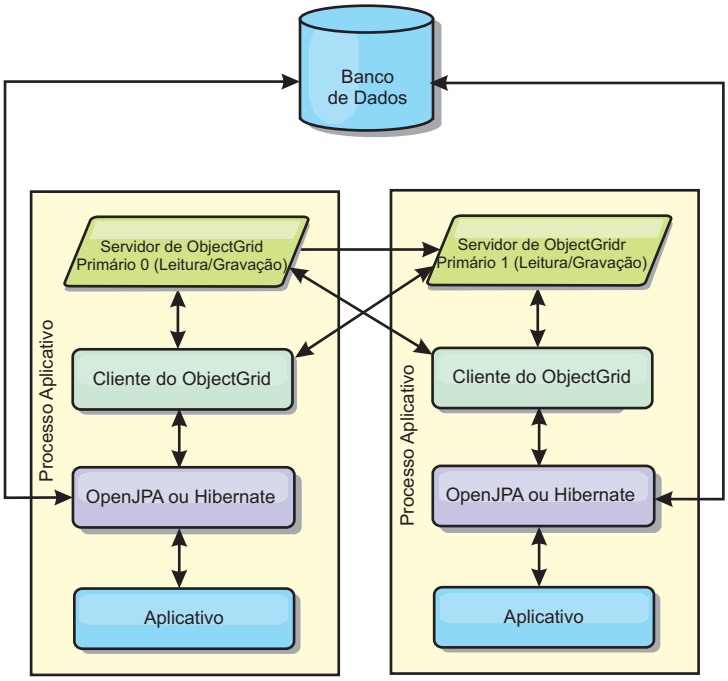


Figura 3. Topologia Particionada Integrada do JPA

Vantagens:

- Armazena grandes quantidades de dados.
- Simples para configurar
- As atualizações de cache são propagadas para vários processos.

Limitação:

- A maioria das leituras e atualizações de cache é remota.

Por exemplo, para armazenar em cache 10 GB de dados com no máximo 1 GB por JVM, dez Java Virtual Machines serão necessários. Portanto, o número de partições deve ser configurado para 10 ou mais. De forma ideal, o número de partições deve ser definido para um número primo no qual cada shard armazena uma quantidade razoável de memória. Geralmente, a configuração de numberOfPartitions é igual ao número de Java Virtual Machines. Com esta configuração, cada JVM armazena uma partição. Se você ativar a replicação, você deve aumentar o número de Java Virtual Machines no sistema; caso contrário, cada JVM também armazena uma partição de réplica, que consome tanta memória quando uma partição primária.

Por exemplo, em um sistema com 4 Java Virtual Machines, e o valor da configuração de numberOfPartitions de 4, cada JVM hospeda uma partição primária. Uma operação de leitura tem 25% mais chance de buscar dados de uma partição disponível localmente, o que é muito mais rápido comparado ao obter dados de uma JVM remota. Se uma operação de leitura, como a execução de uma consulta, precisar buscar uma coleta de dados que envolva 4 partições igualmente, 75% das chamadas são remotas e 25% das chamadas são locais. Se a configuração de ReplicaMode for definida para `SYNC` ou `ASYNC` e a configuração de ReplicaReadEnabled for definida para `true`, as quatro partições de réplica são

criadas e espalhadas pelos quatro Java Virtual Machines. Cada JVM hospeda uma partição primária e uma partição de réplica. A chance de que a operação de leitura execute localmente aumenta em 50%. A operação de leitura que busca uma coleta de dados que envolve quatro partições igualmente tem somente 50% de chamadas remotas e 50% de chamadas locais. Chamadas locais são muito mais rápidas do que chamadas remotas. Sempre que ocorrem chamada remotas, o desempenho cai.

Topologia Remota

Uma topologia remota armazena todos os dados armazenados em cache em um ou mais processos separados, reduzindo o uso da memória dos processos do aplicativo. É possível configurar o eXtreme Scale para ser particionado e replicado. Uma configuração remota é gerenciada independentemente a partir do aplicativo e do provedor JPA.

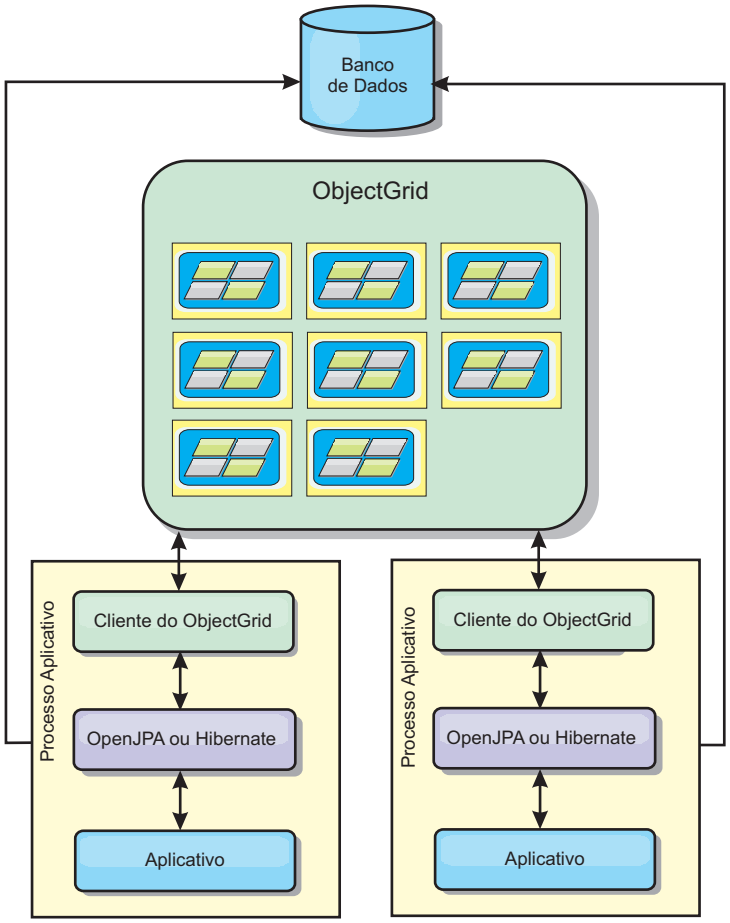


Figura 4. Topologia Remota do JPA

Vantagens:

- Armazena grandes quantidades de dados.
- O processo aplicativo é livre de dados em cache.
- As atualizações de cache são propagadas para vários processos.
- Opções de configuração muito flexíveis.

Limitação:

- Todas as leituras e atualizações de cache são remotas.

Configuração do Plug-in do Cache JPA

O WebSphere eXtreme Scale inclui plug-ins de cache de nível 2 para ambos os provedores, OpenJPA e Hibernate Java Persistence API (JPA).

Propriedades de Configuração do Cache JPA do ObjectGrid

O plug-in do cache JPA do ObjectGrid pode ser configurado com as seguintes propriedades, sendo que todas são opcionais.

ObjectGridName

Especifica o nome exclusivo do ObjectGrid. O valor padrão é o nome da unidade de persistência definido. Se o nome da unidade de persistência não estiver disponível a partir do provedor JPA, um nome gerado será usado.

ObjectGridType

Especifica o tipo de ObjectGrid.

Valores válidos:

- **EMBEDDED**: O tipo de configuração padrão e recomendado. As configurações padrão incluem: `NumberOfPartitions=1`, `ReplicaMode=SYNC`, `ReplicaReadEnabled=true` e `MaxNumberOfReplicas=47`. Use o parâmetro **ReplicaMode** para configurar o modo de replicação e o parâmetro **MaxNumberOfReplicas** para configurar o número máximo de réplicas. Se um sistema tiver mais de 47 Java Virtual Machines, configure o valor **MaxNumberOfReplicas** para que seja igual ao número de Java Virtual Machines.
- **EMBEDDED_PARTITION**: O tipo a ser usado quando o sistema precisar armazenar em cache uma grande quantidade de dados em um sistema distribuído. O número padrão de partições é `47` com um modo de réplica de `NONE`. Em um sistema pequeno com apenas alguns Java Virtual Machines, configure o valor **NumberOfPartitions** para que seja igual ou menor que o número de Java Virtual Machines. Você pode especificar os valores **ReplicaMode**, **NumberOfPartitions** e **ReplicaReadEnabled** para ajustar o sistema.
- **REMOTE**: O cache tenta se conectar com um ObjectGrid remoto e distribuído a partir do serviço de catálogo.

NumberOfPartitions

Valores válidos: maior ou igual a 1Especifica o número de partições a serem usadas para o cache. Essa propriedade é aplicada quando o valor ObjectGridType for configurado para `EMBEDDED_PARTITION`. O valor padrão é `47`. Para o tipo `EMBEDDED`, o valor **NumberOfPartitions** é sempre 1.

ReplicaMode

Valores válidos: `SYNC/ASYNC/NONE`Especifica o método que é usado para copiar o cache para as réplicas. Essa propriedade é aplicada ao configurar o valor ObjectGridType para `EMBEDDED` ou `EMBEDDED_PARTITION`. O valor padrão é `NONE` para o tipo `EMBEDDED_PARTITION` e `SYNC` para o tipo `EMBEDDED`. Se o valor **ReplicaMode** for configurado para `NONE` para o ObjectGridType `EMBEDDED`, o tipo `EMBEDDED` ainda usará um **ReplicaMode** de `SYNC`.

ReplicaReadEnabled

Valores válidos: `TRUE` ou `FALSE`Quando ativado, os clientes leem a partir das réplicas. Essa propriedade é aplicada para o tipo `EMBEDDED_PARTITION`.

O valor padrão é FALSE para o tipo EMBEDDED_PARTITION. O tipo EMBEDDED sempre é configurado para o valor **ReplicaReadEnabled** para TRUE.

MaxUsedMemory

Valores válidos: TRUE ou FALSEAtiva a liberação das entradas de cache quando a memória se tornar limitada. O valor padrão é TRUE e despeja os dados quando o limite de uso do heap da JVM exceder 70%. É possível modificar a porcentagem do limite de uso do heap JVM padrão ao configurar a propriedade memoryThresholdPercentage no arquivo objectGridServer.properties e colocar esse arquivo no caminho de classe.Para obter mais informações sobre os evictors, consulte as informações sobre os evictors no *Visão Geral do Produto*. Para obter mais informações sobre o arquivo de propriedades do servidor, consulte o *Guia de Administração*.

MaxNumberOfReplicas

Valores válidos: maior ou igual a 1Especifica o número máximo de réplicas a ser usado para o cache. Esse valor á aplicado apenas para o tipo EMBEDDED. Esse número deve ser igual ou maior que o número de Java Virtual Machines em um sistema. O valor padrão é 47.

As propriedades NumberOfPartitions, ReplicaMode, ReplicaReadEnabled e MaxNumberOfReplicas são fatores de implementação do ObjectGrid. As propriedades NumberOfPartitions, ReplicaMode e ReplicaReadEnabled aplicam-se ao tipo EMBEDDED_PARTITION. ReplicaMode e MaxNumberOfReplicas aplicam-se ao tipo EMBEDDED.

Considerações sobre EMBEDDED e EMBEDDED_PARTITION

Os tipos de ObjectGrid integrados utilizam as propriedades de configuração anteriormente descritas para configurar e implementar um conjunto de servidores de contêineres do ObjectGrid e um serviço de catálogo, quando necessário. O ciclo de vida dos contêineres é limitado ao aplicativo JPA e é colocado no caminho de classe do aplicativo. Quando um aplicativo for iniciado, o plug-in detecta ou inicia automaticamente um serviço de catálogo, inicia um contêiner ou se conecta ao serviço de catálogo. Em seguida, o plug-in se comunica com o contêiner ObjectGrid e seus equivalentes que estão em execução em outros processos do servidor de aplicativos que usam a conexão do cliente.

Cada entidade JPA possui um mapa de apoio independente designado para usar o nome da classe da entidade. Cada BackingMap possui os seguintes atributos.

- readOnly="false"
- copyKey="false"
- lockStrategy="NONE"
- copyMode="NO_COPY"

Nota: Quando estiver usando o valor de ObjectGridType EMBEDDED ou EMBEDDED_PARTITION em um ambiente do Java SE, use o método System.exit(0) no final do programa para parar o servidor eXtreme Scale integrado. Caso contrário, o programa poderá parecer não estar respondendo.

Padrões do Valor ObjectGridType

O valor ObjectGridType especifica a topologia na qual o cache ObjectGrid é implementado. O tipo padrão e com melhor desempenho é EMBEDDED. As seguintes seções descrevem as propriedades padrão para cada um dos valores ObjectGridType.

Padrões da topologia do cache JPA do ObjectGrid EMBEDDED

Quando você estiver usando o tipo ObjectGrid `EMBEDDED`, os seguintes valores de propriedade padrão serão usados se você não especificar nenhum valor na configuração:

- **ObjectGridName**: nome da unidade de persistência
- **ObjectGridType**: `EMBEDDED`
- **NumberOfPartitions**: `1` (não é possível alterar quando o tipo de ObjectGrid for `EMBEDDED`)
- **ReplicaMode**: `SYNC`
- **ReplicaReadEnabled**: `TRUE` (não é possível alterar quando o tipo de ObjectGrid for `EMBEDDED`)
- **MaxUsedMemory**: `TRUE`
- **MaxNumberOfReplicas**: `47` (deve ser menor ou igual ao número de Java Virtual Machines em um sistema distribuído)

E necessário especificar um único valor de ObjectGridName para evitar conflitos de nomenclatura. O valor de MaxNumberOfReplicas deve ser igual ou maior que o número total de Java Virtual Machines no sistema.

Topologia de Cache do ObjectGrid REMOTE

O tipo de ObjectGrid REMOTE não requer nenhuma configuração de propriedade porque a política de ObjectGrid e de implementação é definida separadamente a partir do aplicativo JPA. O plug-in do cache JPA conecta-se remotamente a um ObjectGrid remoto existente.

Como toda a interação com o ObjectGrid é remota, essa topologia possui o menor desempenho entre todos os tipos de ObjectGrid.

Considerações e Configuração do Serviço de Catálogo

Ao executar uma topologia `EMBEDDED` ou `EMBEDDED_PARTITION`, o plug-in de cache JPA inicia automaticamente um único serviço de catálogo dentro dos processos de aplicativos se necessário. Em um ambiente de produção, é necessário criar uma grade do serviço de catálogo. Para obter mais informações sobre a definição de um serviço de catálogo, consulte .as informações sobre o serviço de catálogo de alta disponibilidade no *Visão Geral do Produto*

Se você estiver executando dentro de um processo do WebSphere Application Server, o plug-in de cache JPA se conecta automaticamente com o serviço de catálogo ou com a grade de serviço de catálogo que é definida para a célula do WebSphere Application Server. Para obter mais informações sobre a definição de uma grade de serviço de catálogo, consulte as informações sobre o início do serviço de catálogo no *Guia de Administração*.

Se você não estiver executando os servidores dentro de um processo doWebSphere Application Server, os host e as portas da grade de serviço do catálogo serão especificadas usando o arquivo de propriedades chamado `objectGridServer.properties`. Este arquivo deve ser armazenado no caminho de classe do aplicativo e ter a propriedade **catalogServiceEndPoints** definida. A grade do serviço de catálogo é iniciada independentemente dos processos do aplicativo e deve ser iniciada antes que os processos do aplicativo sejam iniciados.

O formato do arquivo `objectGridServer.properties` é:

```
catalogServiceEndPoints=<hostname1>:<port1>,<hostname2>:<port2>
```

Configuração de Plug-in do Cache de Hibernação

Um cache do eXtreme Scale pode ser ativado para Hibernate por meio da definição das propriedades no arquivo de configuração.

Configurações

A sintaxe para configurar a propriedade no arquivo `persistence.xml` é a seguinte:

persistence.xml

```
<property name="hibernate.cache.provider_class"
          value="com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider" />
<property name="hibernate.cache.use_query_cache" value="true"/>
<property name="objectgrid.configuration" value="<property>=<value>,..." />
<property name="objectgrid.hibernate.regionNames" value="<regionName>,.." />
```

A sintaxe para configurar a propriedade no arquivo `hibernate.cfg.xml` é a seguinte:

hibernate.cfg.xml

```
<property name="cache.provider_class">com.ibm.websphere.objectgrid.
           hibernate.cache.ObjectGridHibernateCacheProvider</property>
<property name="cache.use_query_cache">true</property>
<property name="objectgrid.configuration"><property>=<value>,...</property>
<property name="objectgrid.hibernate.regionNames"><regionName>,...</property>
```

A propriedade provider_class é com.ibm.websphere.objectgrid.hibernate.cache.ObjectGridHibernateCacheProvider. Para ativar o cache de consulta, defina o valor para `true` na propriedade use_query_cache. Use a propriedade objectgrid.configuration para especificar as propriedades de configuração do cache do eXtreme Scale.

E necessário especificar um único valor da propriedade ObjectGridName para evitar potenciais conflitos de nomenclatura. As outras propriedades de configuração de cache do eXtreme Scale são opcionais.

A propriedade objectgrid.hibernate.regionNames é opcional e deve ser especificada quando os valores regionNames forem definidos após o cache do eXtreme Scale ser inicializado. Considere o exemplo de uma classe de entidade mapeada para uma regionName com a classe de entidade não-especificada no arquivo `persistence.xml` ou não incluída no arquivo de mapeamento Hibernate. Além disso, suponha que ele possua a anotação Entity. Então, o regionName para esta classe de entidade será resolvido no momento do carregamento da classe quando o cache do eXtreme Scale for inicializado. Outro exemplo é a execução do método Query.setCacheRegion(String regionName) que é executado após a inicialização do cache do eXtreme Scale. Nas situações acima, inclua todos os regionNames possíveis determinados como dinâmico na propriedade objectgrid.hibernate.regionNames para que o cache do eXtreme Scale possa preparar o BackingMaps para todos os regionNames.

A seguir estão exemplos dos arquivos `persistence.xml` e `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>
```

Pré-carregando Dados no Cache do ObjectGrid

É possível usar o método preload da classe ObjectGridHibernateCacheProvider para pré-carregar dados no cache do ObjectGrid para uma classe de entidade.

Exemplo 1

Usando EntityManagerFactory

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("testPU");
ObjectGridHibernateCacheProvider.preload("objectGridName", emf, TargetEntity.class, 100, 100);
```

Exemplo 2

Using SessionFactory

```
org.hibernate.cfg.Configuration cfg = new Configuration();
// use o método addResource, addClass e setProperty da Configuração para preparar a
// configuração necessária para criar o SessionFactor
SessionFactory sessionFactory= cfg.buildSessionFactory();
ObjectGridHibernateCacheProvider.preload("objectGridName", sessionFactory, TargetEntity.class, 100, 100);
```

Nota:

1. Em um sistema distribuído, este mecanismo de pré-carregamento somente pode ser chamado de uma Java virtual machine. O mecanismo de pré-carregamento não pode executar simultaneamente a partir de várias Java Virtual Machines.
2. Antes de executar o pré-carregamento, você deve inicializar o cache doeXtreme Scale por meio da criação do EntityManager usando EntityManagerFactory para ter todos os BackingMaps correspondentes criados; caso contrário, o pré-carregamento força o cache a ser inicializado com somente um BackingMap padrão para suportar todas as entidades. Isto significa que um único BackingMap é compartilhado por todas as entidades.

Customizando a Configuração do Cache Hibernate com XML

Para a maioria dos cenários, a configuração das propriedades de cache devem ser suficientes. Para customizar ainda mais o ObjectGrid usado pelo cache, é possível fornecer os arquivos XML de configuração Hibernate ObjectGrid no diretório `META-INF` da mesma forma que o arquivo `persistence.xml`. Durante a inicialização, o cache tentará localizar esses arquivos XML e processá-los, caso sejam localizados.

Há três tipos de arquivos XML de configuração Hibernate ObjectGrid: `hibernate-objectGrid.xml` (configuração ObjectGrid), `hibernate-objectGridDeployment.xml` (política de implementação) e `hibernate-objectGrid-client-override.xml` (configuração de substituição do ObjectGrid cliente). Dependendo da topologia do eXtreme Scale configurada, é possível fornecer qualquer um destes três arquivos XML para customizar essa topologia.

Para os tipos EMBEDDED e EMBEDDED_PARTITION, é possível fornecer qualquer um dos três arquivos XML para customizar o ObjectGrid, a política de implementação e a configuração de substituição do ObjectGrid cliente.

Para um REMOTE ObjectGrid, o cache não cria um ObjectGrid dinâmico. O cache obtém apenas um ObjectGrid do lado do cliente a partir do serviço do catálogo. É possível fornecer apenas um arquivo `hibernate-objectGrid-client-override.xml` para customizar a configuração de substituição do ObjectGrid do cliente.

1. **Configuração do ObjectGrid**: Use o arquivo `META-INF/hibernate-objectGrid.xml`. Este arquivo é utilizado para customizar a configuração do ObjectGrid para os tipos EMBEDDED e EMBEDDED_PARTITION. Com o tipo REMOTE, este arquivo é ignorado. Por padrão, cada classe de entidade possui uma regionName associada (padrão para o nome da classe de entidade) que é mapeada para uma configuração de BackingMap denominada como regionName na configuração do ObjectGrid. Por exemplo, a classe de entidade com.mycompany.Employee possui uma regionName associada definida por padrão como com.mycompany.Employee BackingMap. A configuração padrão do BackingMap é readOnly=″false″, copyKey=″false″, lockStrategy=″NONE″ e `copyMode="NO_COPY"`. É possível customizar alguns BackingMaps com uma configuração escolhida. A palavra-chave reservada ″ALL_ENTITY_MAPS″ pode ser utilizada para representar todos os mapas, exceto outros mapas customizados listados no arquivo `hibernate-objectGrid.xml`. Os BackingMaps que não estiverem listados neste arquivo `hibernate-objectGrid.xml` utilizarão a configuração padrão.
2. **Configuração do ObjectGridDeployment**: Use o arquivo `META-INF/hibernate-objectGridDeployment.xml`. Este arquivo é utilizado para customizar a política de implementação. Ao customizar a política de implementação, se o arquivo `hibernate-objectGridDeployment.xml` for fornecido, a política de implementação padrão será descartada. Todos os valores de atributo da política de implementação serão fornecidos pelo arquivo `hibernate-objectGridDeployment.xml`.
3. **Configuração de substituição do ObjectGrid do cliente**: Use o arquivo `META-INF/hibernate-objectGrid-client-override.xml`. Este arquivo é utilizado para customizar um ObjectGrid do lado do cliente. Por padrão, o cache do ObjectGrid aplica uma configuração de substituição do cliente padrão que desativa o cache local. Se o aplicativo requerer um cache perto, ele pode fornecer este arquivo e especificar `numberOfBuckets=″xxx″`. A substituição do cliente padrão desativa o cache próximo ao configurar `numberOfBuckets=″0″`. O cache perto pode estar ativo ao reconfigurar numberOfBuckets com um valor superior a 0 através de `hibernate-objectGrid-client-override.xml`. A maneira

como o arquivo hibernate-objectGrid-client-override.xml funciona é similar ao hibernate-objectGrid.xml: ele substitui ou estende a configuração de substituição do ObjectGrid do cliente padrão.

Exemplos de Arquivo XML Hibernate ObjectGrid

Os arquivos XML Hibernate ObjectGrid devem ser criados com base na configuração de uma unidade de persistência.

Um arquivo persistence.xml de exemplos que representa a configuração de uma unidade de persistência está a seguir:

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>
```

A seguir está o arquivohibernate-objectGrid.xml que corresponde ao arquivo persistence.xml:

hibernate-objectGrid.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="Annuity">
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <backingMap name="defaultCacheMap" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="defaultCacheMap" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
      <backingMap name="org.hibernate.cache.UpdateTimestampsCache" readOnly="false" copyKey="false"
```

```
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="org.hibernate.cache.UpdateTimestampsCache" />
      <backingMap name="org.hibernate.cache.StandardQueryCache" readOnly="false" copyKey="false"
                lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                pluginCollectionRef="org.hibernate.cache.StandardQueryCache" />
    </objectGrid>
  </objectGrids>
  <backingMapPluginCollections>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="defaultCacheMap">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="org.hibernate.cache.UpdateTimestampsCache">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>


    </backingMapPluginCollection>
    <backingMapPluginCollection id="org.hibernate.cache.StandardQueryCache">
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Nota: Os mapas org.hibernate.cache.UpdateTimestampsCache, org.hibernate.cache.StandardQueryCache e defaultCacheMap são necessários.

A seguir está o arquivo `hibernate-objectGridDeployment.xml` que corresponde ao `persistence.xml`:

hibernate-objectGridDeployment.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="Annuity">
    <mapSet name="MAPSET_Annuity" numberOfPartitions="1" numInitialContainers="1" minSyncReplicas="0"
          maxSyncReplicas="4" maxAsyncReplicas="0" replicaReadEnabled="true">
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <map ref="defaultCacheMap" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
      <map ref="org.hibernate.cache.UpdateTimestampsCache" />
      <map ref="org.hibernate.cache.StandardQueryCache" />
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

Nota: Os mapas org.hibernate.cache.UpdateTimestampsCache, org.hibernate.cache.StandardQueryCache e defaultCacheMap são necessários.

Sistema Externo para um Cache com o Tipo REMOTE do ObjectGrid

É necessário configurar um sistema eXtreme Scale externo se desejar configurar um cache com um tipo de ObjectGrid REMOTE. É necessário ambos os arquivos XML de configuração ObjectGrid e ObjectGridDeployment, que se baseiam no arquivo `persistence.xml`, para configurar um sistema externo. Os arquivos XML de configuração Hibernate de ObjectGrid e ObjectGridDeployment descritos na seção de exemplo de arquivo XML Hibernate do ObjectGrid também podem ser usados para configurar um sistema eXtreme Scale externo.

Um sistema externo do ObjectGrid possui processos de serviço do catálogo e do servidor ObjectGrid. É necessário iniciar um serviço de catálogos antes de iniciar servidores de contêiner. Consulte os detalhes sobre como iniciar servidores eXtreme Scale independentes e processos de contêiner no *Guia de Administração* para obter informações adicionais.

Resolução de Problemas

1. **CacheException: Falha ao obter o servidor ObjectGrid**

   Com um ObjectGridType EMBEDDED ou EMBEDDED_PARTITION, o cache tentará obter uma instância do servidor do tempo de execução do eXtreme Scale. Em um ambiente Java Platform, Standard Edition, um servidor eXtreme Scale com serviço de catálogo integrado será iniciado. O serviço de catálogo integrado tentará atender na porta 2809. Se esta porta estiver sendo utilizada por outro processo, ocorrerá um erro. Se terminais de serviço de catálogo externo forem especificados, por exemplo, com o arquivo `objectGridServer.properties`, este erro ocorrerá se o nome do host ou a porta for especificado incorretamente.

2. **CacheException: Falha ao obter o REMOTE ObjectGrid para o REMOTE ObjectGrid configurado. objectGridName = [ObjectGridName], Nome da PU = [persistenceUnitName]**

   Este erro ocorre quando o cache falha ao obter um ObjectGrid dos terminais de serviço do catálogo fornecido. Normalmente, o erro é causado por um nome do host ou porta incorreto.

3. **CacheException: Não é possível ter duas PUs [persistenceUnitName_1, persistenceUnitName_2] configuradas com o mesmo ObjectGridName [ObjectGridName] do EMBEDDED ObjectGridType**

   Essa exceção ocorrerá se você tiver uma configuração de muitas unidades de persistência e os caches dessas unidades estiverem forem configurados com o mesmo nome do ObjectGrid e ObjectGridType EMBEDDED. Estas configurações de unidade de persistência podem estar no mesmo arquivo `persistence.xml` ou diferente. É necessário verificar se o nome do ObjectGrid é exclusivo para cada unidade de persistência quando o ObjectGridType for EMBEDDED.

4. **CacheException: REMOTE ObjectGrid [ObjectGridName] não inclui os BackingMaps necessários [mapName_1, mapName_2,...]**

   Com um tipo de ObjectGrid REMOTE, se o ObjectGrid do lado do cliente obtido não tiver BackingMaps de entidade completos para suportar o cache da unidade de persistência, ocorrerá esta exceção. Por exemplo, cinco classes de entidade são listadas na configuração da unidade de persistência, mas o ObjectGrid obtido possui apenas dois BackingMaps. Embora o ObjectGrid obtido possa ter dez BackingMaps, se algum destes cinco BackingMaps de entidade necessários não for localizado nos dez BackingMaps, esta exceção ainda ocorrerá.

Configuração de Plug-in do Cache OpenJPA

O WebSphere eXtreme Scale fornece tanto as implementações DataCache quanto QueryCache para OpenJPA. Em resumo, o cache ObjectGrid do OpenJPA ou o cache do ObjectGrid é um termo comum para as implementações DataCache e QueryCache.

Configurações

O cache do eXtreme Scale é ativado ou desativado para OpenJPA através da configurações das propriedades de configuração openjpa.DataCache e openjpa.QueryCache no arquivo `persistence.xml`. A sintaxe para configuração da propriedade é a seguinte:

```
<property name="openjpa.DataCache"
          value="<object_grid_datacache_class(<property>=<value>,...)"/>
<property name="openjpa.QueryCache"
          value="<object_grid_querycache_class(<property>=<value>,...)"/>
```

Tanto DataCache quanto QueryCache pode pegar as propriedades do cache do eXtreme Scale para configurar o cache usado pela unidade de persistência.

Além da configuração do cache do eXtreme Scale, a propriedade openjpa.RemoteCommitProvider deve ser configurada como `sjvm`:

```
<property name="openjpa.RemoteCommitProvider" value="sjvm"/>
```

O valor de tempo limite especificado com a anotação @DataCache para cada classe de entidade é transportado para o BackingMap ao qual cada entidade é armazenada em cache. Porém, o valor de nome especificado com a anotação @DataCache é ignorado pelo cache do eXtreme Scale. O nome da classe de entidade completamente qualificado é o nome do mapa do cache.

Os métodos pin e unpin de OpenJPA StoreCache e QueryCache não são suportados e não desempenham uma função.

Você pode executar a propriedade ObjectGridName, a propriedade ObjectGridType e outras propriedades relacionadas à política de implementação simples na lista de propriedades da classe do cache ObjectGrid para customizar a configuração do cache. Este é um exemplo:

```
<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"/>
```

As configurações DataCache e QueryCache são independentes uma da outra. É possível ativar qualquer uma das configurações. Entretanto, se as duas estiverem ativadas, a configuração do QueryCache utilizará a mesma configuração que a do DataCache e suas propriedades de configuração serão descartadas.

Customizando a Configuração do Cache OpenJPA com XML

Para a maioria dos cenários, a configuração das propriedades do cache do eXtreme Scale deve ser suficiente. Para customizar ainda mais o ObjectGrid usado pelo cache, é possível fornecer os arquivos XML de configuração OpenJPA ObjectGrid

no diretório META-INF da mesma forma que o arquivo persistence.xml. Durante a inicialização do cache, o cache do ObjectGrid tentará localizar estes arquivos XML e processá-los, se forem localizados.

Há três tipos de arquivos XML de configuração OpenJPA ObjectGrid: openjpa-objectGrid.xml (configuração ObjectGrid), openjpa-objectGridDeployment.xml (política de implementação) e openjpa-objectGrid-client-override.xml (configuração de substituição do ObjectGrid cliente). Dependendo do tipo de ObjectGrid configurado, é possível fornecer algum destes três arquivos XML para customizar o ObjectGrid.

Para ambos os tipos EMBEDDED e EMBEDDED_PARTITION, é possível fornecer qualquer um dos três arquivos XML para customizar o ObjectGrid, a política de implementação e a configuração de substituição do ObjectGrid cliente.

Para um REMOTE ObjectGrid, o cache ObjectGrid não cria um ObjectGrid dinâmico. Em vez disso, ele obtém um ObjectGrid do lado do cliente a partir do serviço do catálogo. É possível fornecer apenas o arquivo openjpa-objectGrid-client-override.xml para customizar a configuração de substituição do ObjectGrid cliente.

1. **Configuração do ObjectGrid**: Use o arquivo META-INF/openjpa-objectGrid.xml. Este arquivo é utilizado para customizar a configuração do ObjectGrid para os tipos EMBEDDED e EMBEDDED_PARTITION. Com o tipo REMOTE, este arquivo é ignorado. Por padrão, cada classe de entidade é mapeada para sua própria configuração do BackingMap denominada como um nome de classe de entidade na configuração do ObjectGrid. Por exemplo, a classe de entidade com.mycompany.Employee é mapeada para o BackingMap com.mycompany.Employee. A configuração padrão do BackingMap é readOnly="false", copyKey="false", lockStrategy="NONE" e copyMode="NO_COPY". É possível customizar alguns BackingMaps com uma configuração escolhida. A palavra-chave reservada ALL_ENTITY_MAPS pode ser utilizada para representar todos os mapas, exceto outros mapas customizados listados no arquivo openjpa-objectGrid.xml. Os BackingMaps que não estiverem listados neste arquivo openjpa-objectGrid.xml utilizarão a configuração padrão.Se os BackingMaps customizados não especificarem o atributo ou as propriedades de BackingMaps e estes atributos forem especificados na configuração padrão, os valores de atributo da configuração padrão serão aplicados. Por exemplo, se uma classe de entidade for anotada com timeToLive=30, a configuração padrão do BackingMap para essa entidade terá timeToLive=30. Se o arquivo openjpa-objectGrid.xml customizado também incluir esse BackingMap mas não especificar o valor timeToLive, o BackingMap customizado terá timeToLive=30 por padrão. O arquivo openjpa-objectGrid.xml pretende substituir ou estender a configuração padrão.
2. **Configuração do ObjectGridDeployment**: Use o arquivo META-INF/openjpa-objectGridDeployment.xml. Este arquivo é utilizado para customizar a política de implementação. Ao customizar a política de implementação, se o arquivo openjpa-objectGridDeployment.xml for fornecido, a política de implementação padrão será descartada. Todos os valores de atributo da política de implementação são fornecidos pelo arquivo openjpa-objectGridDeployment.xml.
3. **Configuração de substituição do ObjectGrid do cliente**: Use o arquivo META-INF/openjpa-objectGrid-client-override.xml. Este arquivo é utilizado para customizar um ObjectGrid do lado do cliente. Por padrão, o cache do ObjectGrid aplica uma configuração do ObjectGrid de substituição do cliente que desativa um cache local. Se um aplicativo requerer um cache perto, ele pode fornecer este arquivo e especificar numberOfBuckets="xxx". A substituição

do cliente padrão desativa o cache perto ao configurar numberOfBuckets="0". O cache perto pode estar ativo ao reconfigurar numberOfBuckets com um valor maior que 0 com o arquivo openjpa-objectGrid-client-override.xml. O arquivo openjpa-objectGrid-client-override.xml funciona da mesma forma que o arquivo openjpa-objectGrid.xml: Ele substitui ou estende a configuração padrão de substituição do ObjectGrid cliente.

Exemplos do Arquivo XML OpenJPA do ObjectGrid

Os arquivos XML OpenJPA do ObjectGrid devem ser criados com base na configuração da unidade de persistência.

A seguir há um arquivo persistence.xml de exemplo que representa a configuração de uma unidade de persistência:

persistence.xml

```
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  version="1.0">
  <persistence-unit name="AnnuityGrid">
    <provider>org.apache.openjpa.persistence.PersistenceProviderImpl</provider>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.FixedAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.EquityAnnuity</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Person</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityHolder</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact</class>
    <class>com.ibm.wssvt.acme.annuity.common.bean.jpa.Address</class>
    <exclude-unlisted-classes>true</exclude-unlisted-classes>

    <properties>
    <!-- Database setting -->


    <!--  enable cache -->
      <property name="openjpa.DataCache"
        value="com.ibm.websphere.objectgrid.openjpa.ObjectGridDataCache(objectGridName=Annuity,
              objectGridType=EMBEDDED, maxNumberOfReplicas=4)" />
      <property name="openjpa.RemoteCommitProvider" value="sjvm" />
      <property name="openjpa.QueryCache"
                value="com.ibm.websphere.objectgrid.openjpa.ObjectGridQueryCache()" />
    </properties>
  </persistence-unit>
</persistence>
```

A seguir há o arquivo openjpa-objectGrid.xml que corresponde ao arquivo persistence.xml:

openjpa-objectGrid.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
  xmlns="http://ibm.com/ws/objectgrid/config">
  <objectGrids>
    <objectGrid name="Annuity">
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject"
                  readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject" />
```

```
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
      <backingMap name="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" evictionTriggers="MEMORY_USAGE_THRESHOLD"
                  pluginCollectionRef="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
      <backingMap name="ObjectGridQueryCache" readOnly="false" copyKey="false"
                  lockStrategy="NONE" copyMode="NO_COPY" pluginCollectionRef="ObjectGridQueryCache"
                  evictionTriggers="MEMORY_USAGE_THRESHOLD"  />
    </objectGrid>
  </objectGrids>
  <backingMapPluginCollections>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection
            id="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout">
      <bean id="ObjectTransformer"
            className="com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer" />
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="ObjectGridQueryCache">
      <bean id="MapIndexPlugin" className="com.ibm.websphere.objectgrid.plugins.index.HashIndex" >
          <property name="Name" type="java.lang.String"
                    value="QueryCacheKeyIndex" description="name of index"/>
          <property name="POJOKeyIndex" type="boolean" value="true" description="POJO Key Index "/>
      </bean>
      <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
      </bean>
    </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

Nota:

1. Cada entidade é mapeada para um BackingMap denominado como o nome completo da classe de entidade.
2. Se as classes de entidade estão em uma hierarquia de herança, as classes-filha serão mapeadas para o BackingMap pai. A hierarquia de herança compartilha um único BackingMap.
3. O mapa ObjectGridQueryCache é necessário para suportar QueryCache.
4. O backingMapPluginCollection para cada mapa de entidade deve ter o ObjectTransformer utilizando a classe com.ibm.ws.objectgrid.openjpa.ObjectGridPCDataObjectTransformer.

5. O backingMapPluginCollection para o mapa ObjectGridQueryCache deve ter o índice de chaves denominado como QueryCacheKeyIndex, conforme mostrado no modelo.
6. O evictor é opcional para cada mapa.

A seguir há o arquivo `openjpa-objectGridDeployment.xml` que corresponde ao arquivo `persistence.xml`:

openjpa-objectGridDeployment.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<deploymentPolicy xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://ibm.com/ws/objectgrid/deploymentPolicy ../deploymentPolicy.xsd"
  xmlns="http://ibm.com/ws/objectgrid/deploymentPolicy">
  <objectgridDeployment objectgridName="Annuity">
    <mapSet name="MAPSET_Annuity" numberOfPartitions="1" numInitialContainers="1"
            minSyncReplicas="0" maxSyncReplicas="4" maxAsyncReplicas="0"
            replicaReadEnabled="true">
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Annuity" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Address" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payor" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Person" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Contact" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.AnnuityPersistebleObject" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Rider" />
      <map ref="com.ibm.wssvt.acme.annuity.common.bean.jpa.Payout" />
      <map ref="ObjectGridQueryCache" />
    </mapSet>
  </objectgridDeployment>
</deploymentPolicy>
```

Nota: O mapa ObjectGridQueryCache é necessário para suportar QueryCache.

Sistema Externo para um Cache com o Tipo REMOTE do ObjectGrid

É necessário configurar um sistema externo se desejar configurar um cache com um tipo de ObjectGrid REMOTE. Você precisa de ambos os arquivos XML de configuração ObjectGrid e ObjectGridDeployment, que se baseiam no arquivo `persistence.xml`, para configurar um sistema externo. Os arquivos XML de configuração OpenJPA ObjectGrid e ObjectGridDeployment, descritos na seção de exemplos de arquivo XML OpenJPA do ObjectGrid, também podem ser usados para configurar um sistema eXtreme Scale externo.

Um sistema eXtreme Scale externo possui ambos os processos de serviço do catálogo e do servidor de contêineres. É necessário iniciar o servidor de catálogos antes de iniciar servidores de contêineres.

Resolução de Problemas

1. **CacheException: Falha ao obter o servidor ObjectGrid**

   Com um ObjectGridType EMBEDDED ou EMBEDDED_PARTITION, o cache do eXtreme Scale tenta obter uma instância do servidor do tempo de execução. Em um ambiente Java Platform, Standard Edition, um servidor eXtreme Scale com serviço de catálogo integrado é iniciado. O serviço de catálogo integrado tentará atender na porta 2809. Se esta porta estiver sendo utilizada por outro processo, ocorrerá um erro. Se terminais de serviço de catálogo externo forem especificados, por exemplo, com o arquivo `objectGridServer.properties`, este erro ocorrerá se o nome do host ou a porta for especificado incorretamente.
2. **CacheException: Falha ao obter o REMOTE ObjectGrid para o REMOTE ObjectGrid configurado. objectGridName = [ObjectGridName], Nome da PU = [persistenceUnitName]**

   Este erro ocorre quando o cache não consegue obter um ObjectGrid dos terminais de serviço de catálogo fornecidos. Normalmente, o erro é causado por um nome do host ou porta incorreto.

3. **CacheException: Não é possível ter duas PUs [persistenceUnitName_1, persistenceUnitName_2] configuradas com o mesmo ObjectGridName [ObjectGridName] do EMBEDDED ObjectGridType**

   Essa exceção ocorrerá se você tiver uma configuração de muitas unidades de persistência e os caches do eXtreme Scale destas unidades forem configurados com o mesmo nome de ObjectGrid e ObjectGridType EMBEDDED. Estas configurações de unidade de persistência podem estar no mesmo arquivo `persistence.xml` ou diferente. É necessário verificar se o nome do ObjectGrid é exclusivo para cada unidade de persistência quando o ObjectGridType for EMBEDDED.

4. **CacheException: REMOTE ObjectGrid [ObjectGridName] não inclui os BackingMaps necessários [mapName_1, mapName_2,...]**

   Com um tipo de ObjectGrid REMOTE, se o ObjectGrid do lado do cliente obtido não tiver BackingMaps de entidade completos para suportar o cache da unidade de persistência, ocorrerá esta exceção. Por exemplo, cinco classes de entidade são listadas na configuração da unidade de persistência, mas o ObjectGrid obtido possui apenas dois BackingMaps. Embora o ObjectGrid obtido possa ter dez BackingMaps, se algum destes cinco BackingMaps de entidade necessários não for localizado nos dez BackingMaps, esta exceção ainda ocorrerá.

Nota: O cache OpenJPA do eXtreme Scale alterou o formato de dados para aumentar o desempenho. Todos os sistemas que estão hospedando aplicativos OpenJPA que são configurados com o eXtreme Scale como um cache L2 devem ser interrompidos antes da migração para o WebSphere eXtreme Scale Versão 7.0.

Suporte de Armazenamento em Cache Write-behind

É possível usar o armazenamento em cache write-behind para reduzir o gasto adicional que ocorre na atualização de um banco de dados backend. As filas de armazenamento em cache write-behind são atualizadas para o plug-in do utilitário de carga.

Apresentação

O armazenamento em cache write-behind enfileira assincronamente as atualizações no plug-in do Utilitário de Carga. É possível melhorar o desempenho desconectando atualizações, inserções e remoções para um mapa, a sobrecarga de atualização do banco de dados de backend. A atualização assíncrona é executada após um atraso baseado em tempo (por exemplo, cinco minutos) ou um atraso baseado em entradas (1000 entradas).

Ao definir a configuração write-behind em um mapa de apoio, um encadeamento write-behind é criado e oculta o utilitário de carga configurado. Quando uma transação do eXtreme Scale insere, atualiza ou remove uma entrada do mapa do eXtreme Scale, um objeto LogElement é criado para cada um destes registros. Estes elementos são enviados para o utilitário de carga write-behind e enfileirados em um ObjectMap especial denominado mapa de fila. Cada mapa de apoio com a configuração write-behind ativada possui seus próprios mapas de fila. Um encadeamento write-behind remove periodicamente os dados enfileirados dos mapas de fila e executa o push deles para o utilitário de carga de backend real.

O utilitário de carga write-behind enviará apenas os tipos insert, update e delete dos objetos LogElement para o utilitário de carga real. Todos os outros tipos de objetos LogElement, por exemplo, o tipo EVICT, são ignorados.

Benefícios

Ativar o suporte write-behind possui os seguintes benefícios:

- Isolamento de falha do backend: O armazenamento em cache write-behind fornece uma camada de isolamento a partir de falhas backend. Quando o banco de dados de backend falha, as atualizações são enfileiradas no mapa de fila. Os aplicativos podem continuar a conduzir transações para o eXtreme Scale. Quando o backend se recupera, os dados no mapa de fila são enviados para o backend.
- Carga de backend reduzida: O utilitário de carga write-behind mescla as atualizações em uma base de chave, portanto, apenas uma atualização mesclada por chave existe no mapa de fila. Esta mesclagem diminui o número de atualizações no backend.
- Desempenho de transação aprimorado: Tempos de transação do eXtreme Scale individuais são reduzidos porque a transação não precisa aguardar até que os dados sejam sincronizados com o backend.

XML Descritor do ObjectGrid

Ao configurar um eXtreme Scale utilizando um arquivo XML descritor doeXtreme Scale, o utilitário de carga write-behind é ativado configurando o atributo writeBehind na tag backingMap. Este é um exemplo:

```
<objectGrid name="library" >
     <backingMap name="book" writeBehind="T300;C900" pluginCollectionRef="bookPlugins"/>
```

No exemplo anterior, o suporte write-behind do mapa de apoio ″book″ é ativado com o parâmetro ″T300;C900″.

O atributo write-behind especifica a duração da atualização máxima e/ou uma contagem máxima de atualização de chave. O formato do parâmetro write-behind é:

```
write-behind attribute  ::= <defaults> | <update time> | <update key count> | <update time> ";" <update key count>
update time  ::= "T" <positive integer>
update key count  ::= "C" <positive integer>
defaults  ::= "" {table}
```

Ocorrem atualizações no utilitário de carga quando um dos seguintes eventos ocorre:

1. O tempo máximo de atualização em segundos decorreu desde a última atualização.
2. O número de chaves atualizadas no mapa de fila alcançou a contagem de chaves de atualização.

Estes parâmetros são apenas dicas. A contagem de atualização real e a duração da atualização estarão dentro do intervalo próximo dos parâmetros. Entretanto, não é garantido que a contagem de atualização real ou a duração da atualização sejam as mesmas que as definidas nos parâmetros. Além disso, a primeira atualização behind pode ocorrer após até o dobro da duração da atualização. Isto é porque o eXtreme Scale escolhe a esmo o horário de início para que todas as partições não acessem o banco de dados simultaneamente.

No exemplo anterior T300;C900, o utilitário de carga grava os dados no backend quando decorrem 300 segundos da última atualização ou quando 900 chaves estão pendentes de atualização.

A duração da atualização padrão é 300 segundos e a contagem de chaves de atualização padrão é de 1000.

A seguinte tabela lista alguns exemplos do atributo write-behind.

Nota: Se você configurar o utilitário de carga write-behind como uma cadeia vazia: `writeBehind=""`, o utilitário de carga write-behind é ativado usando os valores padrão. Portanto, não especifique o atributo writeBehind se não desejar que o suporte write-behind seja ativado.

Tabela 6. Algumas Opções de write-behind

Valor do Atributo	Tempo
T100	A duração da atualização é 100 segundos e a contagem de chaves de atualização é de 1000 (o valor padrão)
C2000	A duração da atualização é de 300 segundos (o valor padrão) e a contagem de chaves de atualização é de 2000.
T300;C900	A duração da atualização é de 300 segundos e a contagem de chaves de atualização é de 900.
""	A duração da atualização é de 300 segundos (o valor padrão) e a contagem de chaves de atualização é de 1000 (o valor padrão).

Ativando o Suporte Write-behind Programaticamente

Quando você cria um mapa de apoio programaticamente para um eXtreme Scale em memória local, é possível usar o método a seguir na interface BackingMap para ativar e desativar o suporte para write-behind.

```
public void setWriteBehind(String writeBehindParam);
```

Para obter detalhes adicionais sobre como usar o método setWriteBehind, consulte "Interface BackingMap" na página 63.

Considerações de Design do Aplicativo

Ativar o suporte write-behind é simples, mas o design de um aplicativo para trabalhar com o suporte write-behind precisa de consideração cuidadosa. Sem o suporte write-behind, a transação do eXtreme Scale engloba a transação de backend. A transação do eXtreme Scale inicia antes do início da transação de backend e termina após a transação de backend terminar.

Com o suporte write-behind ativado, a transação do eXtreme Scale termina antes que a transação de backend inicie. A transação do eXtreme Scale e a transação de backend não estão acopladas.

Limitadores de Integridade Referencial

Cada mapa de apoio que é configurado com suporte write-behind possui seu próprio encadeamento write-behind para enviar os dados para o backend. Portanto, os dados que são atualizados em mapas diferentes em uma transação do eXtreme Scale são atualizados no backend em diferentes transações de backend. Por exemplo, a transação T1 atualiza a chave key1 no mapa Map1 e a chave key2 no mapa Map2. A atualização da key1 para o mapa Map1 é atualizada no backend em uma transação de backend e a key2 atualizada para o mapa Map2 é atualizada no backend em outra transação de backend por encadeamentos write-behind diferentes. Se os dados armazenados no Map1 e Map2 possuírem relações, tais como limitadores de chave estrangeira no backend, as atualizações podem falhar.

Ao projetar os limitadores de integridade referencial em seu banco de dados de backend, certifique-se de que atualizações fora de ordem sejam permitidas.

Atualizações Falhas

Como a transação do eXtreme Scale termina antes de a transação de backend iniciar, é possível que ocorra um falso sucesso da transação. Por exemplo, se você tentar inserir uma entrada em uma transação do eXtreme Scale que não exista no mapa de apoio mais que exista no backend, causando uma chave duplicada, a transação do eXtreme Scale não será bem-sucedida. Entretanto, a transação na qual o encadeamento write-behind insere tal objeto no backend falha com uma exceção de chave duplicada.

Consulte o “Manipulando Atualizações Write-Behind Falhas” na página 120 para saber como manipular essas falhas.

Comportamento do Bloqueio de Mapa de Fila

Outra grande diferença de comportamento da transação é o comportamento de bloqueio. O eXtreme Scale suporta três diferentes estratégias de bloqueio: pessimistic, optimistic e none. Os mapas de fila write-behind utilizam a estratégia de bloqueio pessimista não importando qual estratégia de bloqueio está configurada para seu mapa de apoio. Existem dois diferentes tipos de operações que adquirem um bloqueio no mapa de fila:

- Quando ocorre o commit de uma transação do eXtreme Scale ou acontece um flush (flush de mapa ou flush de sessão), a transação lê a chave no mapa de fila e coloca um bloqueio S na chave.
- Quando ocorre um commit na transação do eXtreme Scale, a transação tenta atualizar o bloqueio S para um bloqueio X na chave.

Devido a este comportamento do mapa de fila extra, é possível visualizar algumas diferenças de comportamento de bloqueio.

- Se o mapa do usuário for configurado como a estratégia de bloqueio PESSIMISTIC, não há muita diferença no comportamento de bloqueio. Sempre que um flush ou commit é chamado, um bloqueio S é colocado na mesma chave no mapa de fila. Durante o momento do commit, um bloqueio X não é adquirido apenas para a chave no mapa do usuário, ele também é adquirido para a chave no mapa de fila.
- Se o mapa do usuário for configurado com a estratégia de bloqueio OPTIMISTIC ou NONE, a transação do usuário seguirá o padrão de estratégia de bloqueio PESSIMISTIC. Sempre que um flush ou commit é chamado, um bloqueio S é adquirido para a mesma chave no mapa de fila. Durante o momento do commit, um bloqueio X é adquirido para a chave no mapa de fila utilizando a mesma transação.

Novas Tentativas de Transações do Utilitário de Carga

O WebSphere eXtreme Scale não suporta transações de 2 fases ou XA. O encadeamento write-behind remove registros do mapa de fila e atualiza os registros no backend. Se o servidor falhar no meio da transação, algumas atualizações de backend podem ser perdidas.

O utilitário de carga write-behind automaticamente tenta gravar novamente as transações falhas e envia um LogSequence em dúvida para o backend para evitar a perda de dados. Esta ação exige que o utilitário de carga seja idempotente, o que significa que, quando o método Loader.batchUpdate(TxId, LogSequence) for chamado duas vezes com o mesmo valor, ele fornecerá o mesmo resultado como se fosse aplicado uma vez. As implementações do utilitário de carga devem

implementar a interface RetryableLoader para ativar este recurso. Consulte na documentação da API para obter detalhes adicionais.

Falha do Utilitário de Carga

O plug-in do utilitário de carga pode falhar quando não consegue se comunicar com o back end do banco de dados. Isto pode acontecer se o servidor de banco de dados ou a conexão de rede estiver inativa. O utilitário de carga write-behind irá enfileirar as atualizações e tentará executar o push das alterações de dados para o utilitário de carga periodicamente. O utilitário de carga deve notificar o tempo de execução do WebSphere eXtreme Scale de que existe um problema de conectividade com o banco de dados gerando uma exceção LoaderNotAvailableException.

Portanto, a implementação do Utilitário de Carga deve poder distinguir uma falha de dados ou uma falha de utilitário de carga físico. A falha de dados deve ser gerada ou gerada novamente como uma exceção LoaderException ou OptimisticCollisionException, mas a falha física do utilitário de carga deve ser gerada ou gerada novamene como uma exceção LoaderNotAvailableException. O WebSphere eXtreme Scale trata estas duas exceções diferentemente:

- Se uma LoaderException for capturada pelo utilitário de carga write-behind, o utilitário de carga write-behind a considerará falha devido a alguma falha de dados, tal como uma falha de chave duplicada. O utilitário de carga write-behind irá remover a atualização do lote e tentará atualizar um registro em um momento para isolar a falha de dados. Se uma exceção LoaderException for capturada novamente durante a atualização de um registro, um registro de atualização falho é criado e registrado no mapa de atualização falho.
- Se uma LoaderNotAvailableException for capturada pelo utilitário de carga write-behind, o utilitário de carga write-behind a considerará falha porque não pode se conectar ao final do banco de dados, por exemplo, porque o backend do banco de dados estiver inativo, uma conexão com o banco de dados não estiver disponível ou a rede estiver inativa. O utilitário de carga write-behind aguardará por 15 segundo e, em seguida, tentará novamente executar uma atualização de lote no banco de dados.

O erro comum é lançar uma LoaderException enquanto uma LoaderNotAvailableException deve ser lançada. Todos os registros enfileirados no utilitário de carga write-behind se tornarão atualizações de registro falhas, o que frustra o propósito do isolamento de falha do backend. Esse erro provavelmente ocorrerá se você gravar um utilitário de carga genérico para se comunicar com os bancos de dados.

O eXtreme Scale fornece o JPALoader como um exemplo. O JPALoader usa a API do JPA para interagir com os backends de banco de dados. Quando a rede falhar, o JPALoader obtém javax.persistence.PersitenceException, mas não reconhece a importância da falha a menos que o estado do SQL e o código de erro do SQL da SQLException em cadeia sejam verificados. O fato de que o JPALoader foi projetado para trabalhar com todos os tipos de banco de dados, complica ainda mais o problema já que os estados e os códigos de erro do SQL são diferentes quando ocorrer uma queda de rede. Para resolver isso, o WebSphere eXtreme Scale fornece uma API ExceptionMapper para que os usuários possam conectar uma implementação para mapear uma Exceção a uma exceção mais consumível. Por exemplo, os usuários podem mapear um javax.persistence.PersitenceException genérica para um LoaderNotAvailableException se o estado ou o código de erro SQL indicar uma queda de rede.

Considerações sobre Desempenho

O suporte ao armazenamento em cache write-behind aumenta o tempo de resposta removendo a atualização do utilitário de carga da transação. Ele também aumenta o rendimento do banco de dados já que as atualizações de banco de dados são combinadas. É importante compreender a sobrecarga introduzida pelo encadeamento write-behind, que executa o pull dos dados do mapa de fila e executa o push para o utilitário de carga.

A contagem máxima de atualização ou o tempo máximo de atualização necessário a ser ajustado com base nos padrões de uso e no ambiente esperados. Se o valor da contagem máxima de atualização ou o tempo máximo de atualização for muito pequeno, o gasto adicional do encadeamento write-behind pode exceder os benefícios. Configurar um valor maior para estes dois parâmetros também pode aumentar o uso da memória para enfileirar os dados e aumentar o tempo de envelhecimento dos registros do banco de dados.

Para obter um melhor desempenho, ajuste os parâmetros write-behind com base nos seguintes fatores:

- Proporção de transações de leitura e gravação
- Mesma frequência de atualização de registro
- Latência de atualização de banco de dados.

Configurando o Suporte Write-behind

Você pode ativar o suporte write-behind usando o arquivo XML descritor do ObjectGrid ou programaticamente usando a interface BackingMap.

Use o arquivo XML descritor do ObjectGrid para ativar o suporte write-behind ou programaticamente usando a interface BackingMap.

Arquivo XML descritor do ObjectGrid

Ao configurar um ObjectGrid usando o arquivo XML descritor do ObjectGrid, o utilitário de carga write-behind é ativado configurando-se o atributo writeBehind na tag backingMap. Este é um exemplo:

```
<objectGrid name="library" >
     <backingMap name="book" writeBehind="T300;C900" pluginCollectionRef="bookPlugins"/>
```

No exemplo anterior, o suporte para write-behind do mapa de apoio book está ativado com o parâmetro T300;C900. O atributo write-behind especifica a duração da atualização máxima e/ou uma contagem máxima de atualização de chave. O formato do parâmetro write-behind é:

```
::= <defaults> | <update time> | <update key count> | <update time> ";"
<update key count>::= "T" <positive integer>::= "C" <positive integer>::= ""
```

- atributo write-behind
- duração da atualização
- quantidade de chaves atualizadas
- defaults

Ocorrem atualizações no utilitário de carga quando um dos seguintes eventos ocorre:

1. O tempo máximo de atualização em segundos decorreu desde a última atualização.

2. O número de chaves atualizadas no mapa de fila alcançou a contagem de chaves de atualização.

Estes parâmetros são apenas dicas. A contagem de atualização real e a duração da atualização estarão dentro do intervalo próximo dos parâmetros. Entretanto, não garantimos que a contagem de atualização real ou a duração da atualização sejam as mesmas que as definidas nos parâmetros. Além disso, a primeira atualização behind pode ocorrer após até o dobro da duração da atualização. Isto ocorre porque ObjectGrid escolhe aleatoriamente o momento de início da atualização para que todas as partições não cheguem no banco de dados simultaneamente.

No exemplo anterior T300;C900, o utilitário de carga grava os dados no backend quando 300 segundos decorreram desde a última atualização ou quando 900 chaves estão pendentes para serem atualizadas. A duração da atualização padrão é 300 segundos e a contagem de chaves de atualização padrão é de 1000.

Manipulando Atualizações Write-Behind Falhas

Como a transação do WebSphere eXtreme Scale é concluída antes da transação de backend iniciar, é possível ter um falso sucesso de transação.

Por exemplo, se você tentar inserir uma entrada em uma transação do eXtreme Scale que não existe no mapa de apoio mas existe no backend, causando uma chave duplicada, a transação do eXtreme Scale será bem-sucedida. Entretanto, a transação na qual o encadeamento write-behind insere tal objeto no backend falha com uma exceção de chave duplicada.

Manipulando atualizações write-behind falhas: lado do cliente

Uma atualização deste tipo, ou qualquer outra atualização de backend falha, é uma atualização write-behind falha. Atualizações write-behind falhas são armazenadas em um mapa de atualização write-behind falho. Este mapa funciona como uma fila de eventos para atualizações falhas. A chave da atualização é um objeto Integer exclusivo e o valor é uma instância do FailedUpdateElement. O mapa de atualização write-behind com falha é configurado com um evictor, que despeja os registros 1 hora depois de ter sido inserido. Assim, os registros de atualização com falha serão perdidos se eles não forem recuperados dentro de 1 hora.

A API do ObjectMap pode ser utilizada para recuperar as entradas do mapa de atualização write-behind falho. O nome do mapa de atualização write-behind falho é: IBM_WB_FAILED_UPDATES_<nome do mapa>. Consulte a documentação da API do WriteBehindLoaderConstants para os nomes de prefixo de cada um dos mapas do sistema write-behind. A seguir há um exemplo.

```
processo com falha - código de exemplo
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);
    // Faça algo interessante com a chave, o valor ou a exceção.

}
session.commit();
```

Uma chamada getNextKey funciona com uma partição específica para cada transação eXtreme Scale. Em um ambiente distribuído, para obter chaves de todas as partições, você deve iniciar múltiplas transações, como mostrado no exemplo a seguir:

```
obtendo chaves de todas as partições - código de exemplo
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);
        // Faça algo interessante com a chave, o valor ou a exceção.

    }
    Session.commit();
}
```

Nota: O mapa de atualização falho fornece uma maneira de monitorar o funcionamento do aplicativo. Se um sistema produzir muitos registros no mapa de atualização falho, isto é um sinal de que o aplicativo ou a arquitetura deve ser reavaliado ou revisado para utilizar o suporte write-behind. A partir da 6.1.0.5, é possível utilizar o script xsadmin para visualizar o tamanho da entrada do mapa de atualização falho.

Manipulando atualizações write-behind falhas: listener do shard

É importante detectar e registrar quando uma transação write-behind falha. Todo aplicativo utilizando write-behind precisa implementar um watcher para manipular atualizações write-behind falhas. Isto evita potencialmente ficar sem memória já os registros no Mapa de atualização inválido não são despejados porque o aplicativo é esperado para manipulá-los.

O código a seguir mostra como plugar um watcher, ou dumper, que deve ser incluído no XML do descritor de ObjectGrid como no fragmento.

```
<objectGrid name="Grid">

        <bean id="ObjectGridEventListener" className="utils.WriteBehindDumper"/>
```

É possível visualizar o bean ObjectGridEventListener que foi incluído, que é o watcher write-behind referido acima. O watcher interage nos Mapas para todos os shards principais em um JVM procurando por aqueles com write-behind ativado. Se ele localizar um, então, ele tenta registrar até 100 atualizações inválidas. Ele continua observando um shard principal até que o shard seja movido para um JVM diferente. Todos os aplicativos utilizando write-behind *devem* utilizar um watcher semelhante a este. Caso contrário, o Java Virtual Machines fica sem memória porque este mapa de erro nunca é despejado.

Consulte Código de amostra de classe do dumper write-behind para obter informações adicionais.

Código de Amostra da Classe do Dumper Write-behind

Essa amostra de código de origem mostra como gravar um watcher (dumper) para manipular atualizações write-behind com falhas.

```
//
//This sample program is provided AS IS and may be used, executed, copied and
//modified without royalty payment by customer (a) for its own instruction and
//study, (b) in order to develop applications designed to run with an IBM
//WebSphere product, either for customer's own internal use or for redistribution
//by customer, as part of such an application, in customer's own products. "
```

```
//
//5724-J34 (C) COPYRIGHT International Business Machines Corp. 2009
//All Rights Reserved * Licensed Materials - Property of 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 Stub de método gerado automaticamente

}

public void initialize(Session arg0) {
 // TODO Stub de método gerado automaticamente

}

public void transactionBegin(String arg0, boolean arg1) {
 // TODO Stub de método gerado automaticamente

}

public void transactionEnd(String arg0, boolean arg1, boolean arg2,
  Collection arg3) {
 // TODO Stub de método gerado automaticamente

}
}
```

Configurando Evictores

Os evictores podem ser configurados usando o arquivo descritor XML ObjectGrid ou programaticamente.

Por Que e Quando Desempenhar Esta Tarefa

Para configurar com o XML, consulte "Arquivo XML descritor do ObjectGrid" na página 150.

Ativando o Evictor TTL

O WebSphere eXtreme Scale fornece um mecanismo padrão para despejar entradas do cache e um plug-in para criar evictors customizados. Um Evictor controla a associação de entradas em cada instância do BackingMap. O Evictor padrão usa uma política de despejo de TTL (Time To Live) para cada instância do BackingMap. Se você fornecer um mecanismo de evictor conectável, geralmente ele utilizará uma política de evicção baseada no número de entradas em vez de baseada no tempo.

Ativando o Evictor TTL Programaticamente

Os evictors TTL estão associados a instâncias do BackingMap. O trecho de código a seguir demonstra como a interface BackingMap pode ser utilizada para configurar os atributos necessários para que, quando uma entrada for criada, ela tenha um tempo de expiração configurado como dez minutos após sua criação.

```
Ativando evictor de tempo de vida programaticamenteimport 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 );
```

O argumento do método setTimeToLive é 600 porque ele indica o valor time to live está em segundos. O código anterior deve ser executado antes de o método initialize ser chamado na instância do ObjectGrid. Estes atributos de BackingMap não podem ser alterados após a inicialização da instância do ObjectGrid. Após a execução do código, qualquer entrada inserida no BackingMap myMap tem um tempo de expiração. Ao final do tempo de expiração, o evictor de TTL remove a entrada.

Se um aplicativo exigir que o tempo de expiração seja configurado como a hora do último acesso mais dez minutos, uma linha no código anterior deverá ser alterada. O argumento que é passado para o método setTtlEvictorType é alterado de TTLType.CREATION_TIME para TTLType.LAST_ACCESS_TIME. Com este valor, o tempo de expiração é calculado como a hora do último acesso mais 10 minutos. Quando uma entrada é criada pela primeira vez, a hora do último acesso é a hora de criação.

Quando o argumento TTLType.LAST_ACCESS_TIME é utilizado, as interfaces ObjectMap e JavaMap podem ser utilizadas para substituir o valor time to live do BackingMap. Este mecanismo permite que um aplicativo utilize um valor time to live diferente para cada entrada criada. Assuma que o fragmento de código precedente tenha sido usado para configurar o atributo ttlType como LAST_ACCESS_TIME e o valor de TTL tenha sido configurado para dez minutos

na instância do BackingMap. Um aplicativo pode então substituir o valor time to live para cada entrada executando o seguinte código antes de criar ou modificar uma entrada:

```
import com.ibm.websphere.objectgrid.Session;
import com.ibm.websphere.objectgrid.ObjectMap;
Session session = og.getSession();
ObjectMap om = session.getMap( "myMap" );
int oldTimeToLive1 = om.setTimeToLive( 1800 );
om.insert("key1", "value1" );
int oldTimeToLive2 = om.setTimeToLive( 1200 );
om.insert("key2", "value2" );
```

No trecho de código anterior, a entrada com a chave key1 tem um tempo de expiração do tempo de inserção mais 30 minutos como resultado da chamada de método setTimeToLive(1800) no ObjectMap. A variável oldTimeToLive1 está configurada como 600 porque o valor time to live do BackingMap será utilizado como um valor padrão se o método setTimeToLive não tiver sido chamado anteriormente no ObjectMap.

A entrada com a chave key2 tem um tempo de expiração do tempo de inserção mais 20 minutos como resultado da chamada de método setTimeToLive(1200) no ObjectMap. A variável oldTimeToLive2 está configurada como 1800 porque o valor time to live da chamada de método ObjectMap.setTimeToLive anterior configurou o time to live como 1800.

O exemplo anterior mostra duas entradas do mapa sendo inseridas no mapa myMap para os valores de chave key1 e key2. Em um ponto no tempo posterior, o aplicativo de um novo encadeamento talvez queira atualizar estas entradas do mapa com novos valores do mapa. No entanto, o aplicativo deseja reter os valores time-to-live utilizados no tempo de inserção para cada entrada do mapa. O exemplo a seguir ilustra como reter os valores de TTL (Time-to-Live), utilizando uma constante definida na interface ObjectMap:

```
Session session = og.getSession();
ObjectMap om = session.getMap( "myMap" );
om.setTimeToLive( ObjectMap.USE_DEFAULT );
session.begin();
om.update("key1", "updated value1" );
om.update("key2", "updated value2" );
om.insert("key3", "value3" );
session.commit();
```

Como o valor especial do ObjectMap.USE_DEFAULT é utilizado na chamada do método setTimeToLive, a chave key1 retém seu valor time to live de 1800 segundos e a chave key2 retém seu valor time to live de 1200 segundos porque tais valores foram utilizados quando estas entradas de mapa foram inseridas pela transação anterior.

O exemplo anterior também mostra uma nova entrada de mapa para a inserção da chave key3. Nesse caso, o valor especial USE_DEFAULT indica a utilização da configuração padrão do valor time-to-live para o mapa. O valor padrão é definido pelo atributo time-to-live de BackingMap. Consulte atributos da interface BackingMap para obter informações sobre como o atributo time-to-live é definido na instância do BackingMap.

Consulte a documentação da API para o método setTimeToLive nas interfaces ObjectMap e JavaMap. A documentação o avisa que uma exceção IllegalStateException resulta se o método BackingMap.getTtlEvictorType() retorna algo que não o valor TTLType.LAST_ACCESS_TIME value. Os mapas ObjectMap e JavaMap só podem ser utilizados para substituírem o valor time-to-live quando o tipo de evictor de TTL utilizado for o LAST_ACCESS_TIME. Esse método não pode ser utilizado para substituir o valor time-to-live quando o tipo de evictor

utilizado for TTL CREATION_TIME ou NONE.

Abordagem de Configuração XML para Ativar o Evictor TTL

Em vez de usar a interface BackingMap para programaticamente configurar os atributos de BackingMap a serem usados pelos Evictor TTL, é possível usar um arquivo XML para configurar cada instância de BackingMap. O código a seguir demonstra como configurar tais atributos para três mapas BackingMap diferentes:

```
Ativando evictor de tempo de vida usando XML
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
<objectGrids>
    <objectGrid name="grid1">
        <backingMap name="map1" ttlEvictorType="NONE" />
        <backingMap name="map2" ttlEvictorType="LAST_ACCESS_TIME" timeToLive="1800" />
        <backingMap name="map3" ttlEvictorType="CREATION_TIME" timeToLive="1200" />
    </objectgrid>
</objectGrids>
```

Esse exemplo mostra que o BackingMap map1 utiliza o tipo de evictor de TTL NONE. O BackingMap map2 utiliza o tipo de evictor de TTL LAST_ACCESS_TIME e tem um valor time-to-live de 1.800 segundos ou 30 minutos. O BackingMap map3 está definido para utilizar um tipo de evictor de TTL CREATION_TIME e tem um valor time-to-live de 1.200 segundos ou 20 minutos.

Evictores Conectáveis Opcionais

O evictor TTL padrão utiliza uma política de evicção baseada no tempo e o número de entradas no BackingMap não tem efeito sobre o tempo de expiração de uma entrada. É possível utilizar um evictor conectável opcional para despejar entradas com base no número de entradas existentes em vez de no tempo.

Os seguintes evictors conectáveis opcionais fornecem alguns algoritmos comumente utilizados para decidir quais entradas liberar quando um BackingMap crescer além de algum limite de tamanho.

- O evictor LRUEvictor utiliza um algoritmo LRU (Least Recently Used) para decidir quais entradas serão despejadas quando o BackingMap exceder um número máximo de entradas.
- O evictor LFUEvictor utiliza um algoritmo LFU (Least Frequently Used) para decidir quais entradas serão despejadas quando o BackingMap exceder um número máximo de entradas.

O BackingMap informa um evictor conforme as entradas são criadas, modificadas ou removidas de uma transação. O BackingMap acompanha estas entradas e escolhe quando liberar uma ou mais entradas da instância do BackingMap.

Uma instância do BackingMap não possui informações de configuração para um tamanho máximo. Em vez disso, as propriedades do evictor são configuradas para controlar o comportamento do evictor. O LRUEvictor e o LFUEvictor possuem uma propriedade de tamanho máximo utilizada para fazer o evictor começar a liberar entradas quando o tamanho máximo for excedido. Assim como o evictor TTL, os evictores LRU e LFU podem não liberar imediatamente uma entrada quando o número máximo de entradas for atingido para minimizar o impacto no desempenho.

Se o algoritmo LRU ou LFU não for adequado para um determinado aplicativo, você poderá redigir seus próprios evictors para criar a sua estratégia de despejo.

Utilizando Evictors Conectáveis Opcionais

Para incluir evictors conectáveis opcionais na configuração do BackingMap, é possível utilizar tanto a configuração programática quanto a configuração do XML.

Conectando um Evictor Conectável Programaticamente

Como os evictors estão associados aos BackingMaps, use a interface BackingMap para especificar o evictor conectável. O trecho de código a seguir é um exemplo de especificação de um evictor LRUEvictor para o BackingMap map1 e um evictor LFUEvictor para a instância do BackingMap map2:

Conectando um evictor programaticamente

```
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);
```

O snippet anterior mostra um evictor LRUEvictor sendo utilizado para o map1 BackingMap com um número aproximado de entradas de 53.000 (53 * 1000). O evictor LFUEvictor é utilizado para o map2 BackingMap com um número máximo aproximado de entradas de 422.000 (211 * 2000). Os evictores LRU e LFU têm uma propriedade de tempo de suspensão que indica por quanto tempo o evictor fica suspenso antes de ser ativado e verificar se as entradas precisam ser liberadas. O tempo de suspensão é especificado em segundos. Um valor de 15 segundos é uma boa garantia entre o impacto no desempenho e a prevenção para que o BackingMap não se torne muito grande. A meta é utilizar o maior tempo de suspensão possível sem fazer o BackingMap crescer a um tamanho excessivo.

O método setNumberOfLRUQueues configura a propriedade LRUEvictor que indica quantas filas de LRU o evictor utiliza para gerenciar informações de LRU. Uma coleta de filas é utilizada para que cada entrada não mantenha informações de LRU na mesma fila. Esta abordagem pode melhorar o desempenho minimizando o número de entradas do mapa que precisam ser sincronizadas no mesmo objeto de fila. Aumentar o número de filas é uma boa maneira de minimizar o impacto que o evictor LRU pode causar no desempenho. Um bom ponto de partida é utilizar dez por cento do número máximo de entradas como o número de filas. A utilização de um número primo geralmente é melhor do que utilizar um número que não seja primo. O método setMaxSize indica quantas entradas são permitidas em cada fila. Quando uma fila alcança seu número máximo entradas, a entrada ou as entradas utilizadas menos recentemente nesta fila são despejadas na próxima vez que o evictor verifica se há alguma entrada que precisa ser despejada.

O método setNumberOfHeaps configura a propriedade LFUEvictor para determinar quantos objetos de heap binários o LFUEvictor utiliza para gerenciar

informações de LFU. Mais uma vez é utilizada uma coleta para melhorar o desempenho. A utilização de dez por cento do número máximo de entrada é um bom ponto de partida e utilizar um número primo geralmente é melhor do que utilizar um número que não seja primo. O método setMaxSize indica quantas entradas são permitidas em cada heap. Quando um heap alcança seu número máximo entradas, a entrada ou as entradas utilizadas menos recentemente neste heap são despejadas na próxima vez que o evictor verifica se há alguma entrada que precisa ser despejada.

Abordagem de Configuração XML para Conectar um Evictor Conectável

Em vez de utilizar várias APIs para conectar programaticamente um evictor e configurar suas propriedades, um arquivo XML pode ser utilizado para configurar cada BackingMap conforme ilustrado na amostra a seguir:

Conectando um evictor usando XML
```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
<objectGrids>
    <objectGrid name="grid">
        <backingMap name="map1" ttlEvictorType="NONE" pluginCollectionRef="LRU" />
        <backingMap name="map2" ttlEvictorType="NONE" pluginCollectionRef="LFU" />
    </objectGrid>
</objectGrids>
<backingMapPluginCollections>
    <backingMapPlugincollection id="LRU">
        <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor">
            <property name="maxSize" type="int" value="1000" description="set max size for each LRU queue" />
            <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
            <property name="numberOfLRUQueues" type="int" value="53" description="set number of LRU queues" />
        </bean>
    </backingMapPluginCollection>
    <backingMapPluginCollection id="LFU">
        <bean id="Evictor" className="com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor">
            <property name="maxSize" type="int" value="2000" description="set max size for each LFU heap" />
            <property name="sleepTime" type="int" value="15" description="evictor thread sleep time" />
            <property name="numberOfHeaps" type="int" value="211" description="set number of LFU heaps" />
        </bean>
    </backingMapPluginCollection>
</backingMapPluginCollections>
</objectGridConfig>
```

Despejo Baseado em Memória

Todos os evictors integrados suportam despejo baseado em memória que pode ser ativado na interface BackingMap configurando o atributo evictionTriggers de BackingMap como ″MEMORY_USAGE_THRESHOLD″. Para obter mais informações sobre como configurar o atributo evictionTriggers no BackingMap, consulte a interface BackingMap e a referência de configuração do eXtreme Scale.

O despejo baseado em memória é baseado no limite de uso do heap. Quando o despejo baseado em memória é ativado no BackingMap e o BackingMap possui algum evictor integrado, o limite de uso é configurado com uma porcentagem padrão de memória total se o limite ainda não tiver sido configurado anteriormente.

Para alterar a porcentagem de limite de uso, configure a propriedade memoryThresholdPercentage no contêiner e o arquivo de propriedades do servidor para o processo do servidor do eXtreme Scale. Para configurar o limite de uso de destino em um processo do cliente do eXtreme Scale, é possível utilizar o MemoryPoolMXBean. Consulte também: Arquivo containerServer.props e Iniciando Processos do Servidor eXtreme.

Durante o tempo de execução, se o uso da memória exceder o limite de uso destinado, os evictors baseados em memória iniciam o despejo de entradas e tentam manter o uso da memória abaixo do limite de uso destinado. Entretanto, não há garantia de que a velocidade do despejo seja rápida o suficiente para evitar um potencial erro de falta de memória se o tempo de execução do sistema continuar a consumir memória rapidamente.

Conexão de um Evictor programável

Como os evictors são associados com BackingMaps, utilize a interface BackingMap para especificar o evictor conectável.

Conectando um Evictor Conectável Programaticamente

O trecho de código a seguir é um exemplo de especificação de um evictor LRUEvictor para o BackingMap map1 e um evictor LFUEvictor para o 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);
```

O snippet anterior mostra um evictor LRUEvictor sendo utilizado para o map1 BackingMap com um número aproximado de entradas de 53.000 (53 * 1000). O evictor LFUEvictor é utilizado para o map2 BackingMap com um número máximo aproximado de entradas de 422.000 (211 * 2000). Os evictores LRU e LFU têm uma propriedade de tempo de suspensão que indica por quanto tempo o evictor fica suspenso antes de ser ativado e verificar se as entradas precisam ser liberadas. O tempo de suspensão é especificado em segundos. Um valor de 15 segundos é uma boa garantia entre o impacto no desempenho e a prevenção para que o BackingMap não se torne muito grande. A meta é utilizar o maior tempo de suspensão possível sem fazer o BackingMap crescer a um tamanho excessivo.

O método setNumberOfLRUQueues configura a propriedade LRUEvictor que indica quantas filas de LRU o evictor utiliza para gerenciar informações de LRU. Uma coleta de filas é utilizada para que cada entrada não mantenha informações de LRU na mesma fila. Esta abordagem pode melhorar o desempenho minimizando o número de entradas do mapa que precisam ser sincronizadas no mesmo objeto de fila. Aumentar o número de filas é uma boa maneira de minimizar o impacto que o evictor LRU pode causar no desempenho. Um bom ponto de partida é utilizar dez por cento do número máximo de entradas como o número de filas. A utilização de um número primo geralmente é melhor do que utilizar um número que não seja primo. O método setMaxSize indica quantas

entradas são permitidas em cada fila. Quando uma fila alcança seu número máximo entradas, a entrada ou as entradas utilizadas menos recentemente nesta fila são despejadas na próxima vez que o evictor verifica se há alguma entrada que precisa ser despejada.

O método setNumberOfHeaps configura a propriedade LFUEvictor para determinar quantos objetos de heap binários o LFUEvictor utiliza para gerenciar informações de LFU. Mais uma vez é utilizada uma coleta para melhorar o desempenho. A utilização de dez por cento do número máximo de entrada é um bom ponto de partida e utilizar um número primo geralmente é melhor do que utilizar um número que não seja primo. O método setMaxSize indica quantas entradas são permitidas em cada heap. Quando um heap alcança seu número máximo entradas, a entrada ou as entradas utilizadas menos recentemente neste heap são despejadas na próxima vez que o evictor verifica se há alguma entrada que precisa ser despejada.

Abordagem de Configuração XML para Conectar um Evictor Conectável

Em vez de utilizar várias APIs para conectar programaticamente um evictor e configurar suas propriedades, um arquivo XML pode ser utilizado para configurar cada BackingMap conforme ilustrado na amostra a seguir:

```
<?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>
```

Configurando o HashIndex

Este plug-in HashIndex suporta ambas as interfaces MapIndex e MapRangeIndex. Definir e utilizar índices adequadamente pode aprimorar significativamente o desempenho da consulta.

Os seguintes atributos podem ser usados para configurar o HashIndex usando o arquivo XML descritor de implementação do ObjectGrid ou uma abordagem programática:

- Nome: O nome do índice. O nome deve ser exclusivo para cada mapa. O nome é usado para recuperar o objeto do índice da instância de ObjectMap para o BackingMap.

- AttributeName: Os nomes delimitados por vírgula dos atributos para índice. Para índices acessados por campo, os nomes de atributos são equivalentes aos nomes de campo. Para índices acessados por propriedade, os nomes de atributos são os nomes da propriedade compatível com JavaBean. Se existir somente um nome de atributo, o HashIndex é um índice de atributo único, e se este atributo for um relacionamento, ele também é um índice de relacionamento. Se múltiplos nomes de atributo são incluídos nos nomes de atributo, o HashIndex é um índice composto.
- FieldAccessAttribute: Usado para mapas sem entidade. Se true, o objeto é acessado usando os campos diretamente. Se não especificado ou false, o método getter do atributo é usado para acessar os dados.
- POJOKeyIndex: Usado para mapas sem entidade. Se true, o índice introspectará o objeto na parte da chave do mapa. Isto é útil quando a chave é uma chave composta e o valor não tem a chave integrada dentro dele. Se não especificado ou false, então o índice introspectará o objeto na parte do valor do mapa.
- RangeIndex: Se true, a indexação de intervalo é ativada e o aplicativo pode converter o objeto do índice recuperado para a interface de MapRangeIndex. Se a propriedade RangeIndex for configurada como “false”, o aplicativo pode somente converter o objeto do índice recuperado para a interface MapIndex.

HashIndex de Atributo Único versus HashIndex Composto

Quando a propriedade AttributeName de HashIndex inclui múltiplos nomes de atributos, o HashIndex é um índice composto. Caso contrário, se ela incluir somente um nome de atributo, ela é um índice de atributo único. Por exemplo, o valor da propriedade AttributeName de um HashIndex composto pode ser ″city,state,zipcode”. Ela inclui três atributos delimitados por vírgulas. Se o valor da propriedade AttributeName for somente “zipcode”, que tem apenas um atributo, ela é um HashIndex de atributo único.

O HashIndex composto fornece uma maneira eficiente de consultar objetos em cache quando os critérios de busca envolvem muitos atributos. Porém, ele não suporta o índice de intervalo e sua propriedade RangeIndex deve ser definida para “false”.

Consulte o tópico sobre HashIndex composto no *Guia de Administração*.

HashIndex de Relacionamento

Se o atributo indexado de um HashIndex de atributo único for um relacionamento, tanto com valor único ou múltiplos valores, o HashIndex é um HashIndex de relacionamento. Para HashIndex de relacionamento, a propriedade RangeIndex de HashIndex deve ser definida para “false”.

O HashIndex de relacionamento pode acelerar consultas que usam referência cíclicas ou usar filtros de consulta IS NULL, IS EMPTY, SIZE e MEMBER OF. Consulte otimização de consulta usando índices no *Guia de Programação*.

HashIndex de Intervalo

Quando a propriedade RangeIndex de HashIndex está definida para “true”, o HashIndex é um índice de intervalo e pode suportar a interface MapRangeIndex. UmMapRangeIndex suporta funções para localizar dados usando funções de intervalo, como maior que, menor que, ou ambas, enquanto um MapIndex suporta apenas funções iguais. Para um índice com atributo único, a propriedade

RangeIndex pode ser definida para “true” somente se o atributo indexado for do tipo Comparable. Se o índice de atributo único for usado pela consulta, a propriedade RangeIndex deverá ser definida para “true” e o atributo indexado deverá ser do tipo Comparable. Para HashIndex de relacionamento e HashIndex composto, a propriedade RangeIndex deve ser definida para “false”.

A seguir está um resumo para o uso do índice de intervalo.

Tabela 7. Suporte para Índice de Intervalo

Tipo HashIndex	**Suporta índice de intervalo**
HashIndex de atributo único: chave ou atributo indexado é do tipo Comparable	Yes
HashIndex de atributo único: chave ou atributo indexado não é do tipo Comparable	Não
HashIndex Composto	Não
HashIndex de Relacionamento	Não

Otimização de Consulta Utilizando HashIndex

Definir e utilizar índices adequadamente pode aprimorar significativamente o desempenho da consulta. As consultas do WebSphere® eXtreme Scale podem usar plug-ins HashIndex integrados para melhorar o desempenho das consultas. Ainda que o uso do índice possa melhorar significantemente o desempenho da consulta, ele por ter um impacto no desempenho em operações de mapa transacional.

Configurando Replicação Ponto a Ponto com o JMS

Mecanismo de replicação ponto a ponto baseado no Java Message Service (JMS) é usado em ambiente local e distribuído do WebSphere eXtreme Scale. O JMS é um processo de replicação núcleo a núcleo e permite que as atualizações de dados fluam entre os ObjectGrids e os ObjectGrids distribuídos. Por exemplo, com esse mecanismo é possível mover as atualizações de dados de uma grade distribuída do eXtreme Scale para uma grade local do eXtreme Scale ou de uma grade para outra grade em um domínio de sistema diferente.

Antes de Iniciar

O mecanismo de replicação ponto a ponto baseado em JMS é baseado no ObjectGridEventListener baseado em JMS integrado, com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener. Para obter informações detalhadas sobre a ativação do mecanismo de replicação ponto a ponto, consulte “Listener de Eventos da JMS” na página 139.

Consulte “Ativando o Mecanismo de Invalidação do Cliente” na página 77 para obter informações adicionais.

A seguir há um exemplo de uma configuração XML para ativar um mecanismo de replicação ponto a ponto em uma configuração do eXtreme Scale:

Configuração de replicação ponto a ponto - exemplo de XML
```
<bean id="ObjectGridEventListener"
classNam="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>
```

Ativando o Mecanismo de Invalidação do Cliente

Em um ambiente do WebSphere eXtreme Scale distribuído, o lado do cliente possui um cache local por padrão ao utilizar a estratégia de bloqueio otimista ou quando o bloqueio está desativado. O cache perto tem seus próprios dados locais armazenados em cache. Se um eXtreme Scale cliente executar uma atualização, esta atualização chegará até seu cache perto e servidor do cliente. Entretanto, outros eXtreme Scale clientes não recebem as informações de atualização e poderão ter dados desatualizados.

Cache Local

Os aplicativos devem ficar cientes desse problema de dados obsoletos no cliente do eXtreme Scale. É possível usar a classe ObjectGridEventListener baseada em Java Message Service (JMS) integrado, com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener, para ativar o mecanismo de invalidação do cliente dentro de um ambiente do eXtreme Scale distribuído que é conhecido como uma grade eXtreme Scale.

O mecanismo de invalidação do cliente é a solução para o problema de dados antigos no cache local do cliente no ambiente do eXtreme Scale distribuído. Esse mecanismo garante que o cache local ao cliente esteja em sincronia com os servidores ou outros clientes. Contudo, mesmo com esse mecanismo de invalidação do cliente baseado em JMS, o cache local do cliente não é atualizado imediatamente. Um atraso ocorre quando o tempo de execução do eXtreme Scale publica atualizações.

Dois modelos estão disponíveis para o mecanismo de invalidação do cliente em um ambiente do eXtreme Scale distribuído:

- Modelo cliente-servidor: Neste modelo, todos os processos do servidor assumem a função de publicador e publicam todas as alterações na transação para o destino do JMS designado. Todos os processos do cliente estão nas funções de receptor e recebem todas as alterações na transação do destino do JMS designado.
- Modelo cliente como dupla função: Neste modelo, os processos do servidor não têm nada a fazer com o destino do JMS. Todos os processos do cliente assumem ambas as funções do JMS, publicador e receptor. As alterações na transação que ocorrem no cliente são publicadas no destino do JMS e todos os clientes recebem tais alterações.

Para obter mais informações sobre a ativação do mecanismo de invalidação do cliente, consulte “Listener de Eventos da JMS” na página 139.

Modelo de Cliente-servidor

Em um modelo cliente/servidor, os servidores exercem a função de publicador JMS e o cliente a função de receptor JMS.

Exemplo de XML de modelo cliente/servidor

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml version="1.0" encoding="UTF-8"?>
```

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

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

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

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

</objectGridConfig>
```

Cliente como Modelo de Funções Duplas

No modelo cliente como dupla função, cada cliente assume ambas as funções do JMS, publicador e receptor. O cliente publica cada alteração da transação confirmada para um destino JMS designado e recebe todas as alterações transacionais confirmadas de outros clientes. O servidor não tem nada a fazer com o JMS neste modelo.

Exemplo de XML de modelo de função dupla

```
<?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>
```

Distribuição de Alterações Entre Java Virtual Machines Peer

Os objetos LogSequence e LogElement comunicam as alterações que ocorreram em uma transação do eXtreme Scale com um plug-in do ObjectGridEventListener.

Para obter mais informações sobre o como o Java Message Service (JMS) pode ser usado para distribuir alterações transacionais, consulte as informações sobre o uso da JMS para distribuir alterações de transação no *Visão Geral do Produto*.

Um pré-requisito é que a instância do ObjectGrid deve ser armazenada em cache pelo ObjectGridManager. Consulte Métodos createObjectGrid para obter mais informações. O valor booleano cacheInstance deve ser configurado como true.

Não é necessário que você implemente esse mecanismo. Há um mecanismo de replicação ponto a ponto integrado disponível para utilizar esta função. Consulte as informações sobre a configuração de replicação ponto a ponto com JMS no *Guia de Administração*.

Os objetos fornecem um meio para que um aplicativo publique facilmente alterações que ocorreram em um ObjectGrid utilizando um transporte de mensagens para ObjectGrids peer nas Java Virtual Machines remotas e, então, aplica estas alterações em tais JVM. A classe LogSequenceTransformer é importante para ativar este suporte. Este artigo examina como escrever um listener usando um sistema de mensagens JVM (Java Message Service) para propagação das mensagens. Para este fim, o eXtreme Scale suporta a transmissão de LogSequences que resultam de uma consolidação transacional de um eXtreme Scale através dos membros do cluster do WebSphere Application Server com um plug-in fornecido pela IBM. Esta função não é ativada por padrão, mas pode ser configurada para

ser operacional. Porém, quando o consumidor ou produtor não for um WebSphere Application Server, o uso de um sistema de mensagens JMS externo pode ser necessário.

Implementando o Mecanismo

A classe LogSequenceTransformer e as APIs ObjectGridEventListener, LogSequence e LogElement permitem que qualquer publicação-e-assinatura confiável seja utilizada para distribuir as alterações e filtrar os mapas que você deseja distribuir. Os snippets neste tópico mostram como utilizar estas APIs com o JMS para criar um ObjectGrid ponto a ponto compartilhado por aplicativos que estão hospedados em um conjunto diverso de plataforma compartilhando um transporte de mensagens comum.

Inicialize o plug-in

O ObjectGrid chama o método de inicialização do plug-in, parte do contrato da interface ObjectGridEventListener, quando o ObjectGrid inicia. O método de inicialização deve obter seus recursos JMS, incluindo conexões, sessões, e publicadores, e inicia o encadeamento que é o listener JMS.

Os exemplos a seguir mostram o método de inicialização:

Exemplo do método initialize
```
public void initialize(Session session) {
        mySession = session;
        myGrid = session.getObjectGrid();
        try {
            if (mode == null) {
                throw new ObjectGridRuntimeException("No mode specified");
            }
            if (userid != null) {
                connection = topicConnectionFactory.createTopicConnection(userid, password);
            } else
                connection = topicConnectionFactory.createTopicConnection();

            // need to start the connection to receive messages.
            connection.start();

            // the jms session is not transactional (false).
            jmsSession = connection.createTopicSession(false, javax.jms.Session.AUTO_ACKNOWLEDGE);
            if(topic == null)
                if (topicName == null) {
                    throw new ObjectGridRuntimeException("Topic not specified");
                } else {
                    topic = jmsSession.createTopic(topicName);
                }
            publisher = jmsSession.createPublisher(topic);
            // iniciar o encadeamento do listener.
            listenerRunning = true;
            listenerThread = new Thread(this);
            listenerThread.start();
        } catch (Throwable e) {
            throw new ObjectGridRuntimeException("Cannot initialize", e);
        }
    }
```

O código para iniciar o encadeamento usa um encadeamento Java 2 Platform, Standard Edition (Java SE). Se você estiver executando um WebSphere Application Server Versão 6.x ou um servidor enterprise WebSphere Application Server Version 5.x, use a interface de programação de aplicativos (API) do bean assíncrono para iniciar este encadeamento de daemon. Também é possível utilizar as APIs comuns. A seguir está um snippet de substituição de exemplo que mostra a mesma ação utilizando um gerenciador de trabalho:

```
// iniciar o encadeamento do listener.
listenerRunning = true;
workManager.startWork(this, true);
```

O plug-in também deve implementar a interface Work em vez da interface Runnable. Também é necessário incluir um método de liberação para configurar a

variável listenerRunning como false. O plug-in deve ser fornecido com uma instância do gerenciador de trabalho em seu construtor ou por injeção, se estiver utilizando um contêiner IoC (Inversion of Control).

Transmita as alterações

A seguir, está um método transactionEnd de amostra para a publicação das alterações locais que são feitas em um ObjectGrid. Esta amostra utiliza o JMS, embora seja possível utilizar qualquer transporte de mensagens que seja capaz de emitir mensagens de publicação-e-assinatura confiáveis.

Exemplo do método transactionEnd

```
// This method is synchronized to make sure the
    // messages are published in the order the transaction
    // were committed. If we started publishing the messages
    // in parallel then the receivers could corrupt the Map
    // as deletes may arrive before inserts etc.
    public synchronized void transactionEnd(String txid, boolean isWriteThroughEnabled, boolean committed,
            Collection changes) {
        try {
            // must be write through and commited.
            if (isWriteThroughEnabled && committed) {
                // write the sequences to a byte []
                ByteArrayOutputStream bos = new ByteArrayOutputStream();
                ObjectOutputStream oos = new ObjectOutputStream(bos);
                if (publishMaps.isEmpty()) {
                    // serialize the whole collection
                    LogSequenceTransformer.serialize(changes, oos, this, mode);
                } else {
                    // filter LogSequences based on publishMaps contents
                    Collection publishChanges = new ArrayList();
                    Iterator iter = changes.iterator();
                    while (iter.hasNext()) {
                        LogSequence ls = (LogSequence) iter.next();
                        if (publishMaps.contains(ls.getMapName())) {
                            publishChanges.add(ls);
                        }
                    }
                    LogSequenceTransformer.serialize(publishChanges, oos, this, mode);
                }
                // make an object message for the changes
                oos.flush();
                ObjectMessage om = jmsSession.createObjectMessage(bos.toByteArray());
                // set properties
                om.setStringProperty(PROP_TX, txid);
                om.setStringProperty(PROP_GRIDNAME, myGrid.getName());
                // transmit it.
                publisher.publish(om);
            }
        } catch (Throwable e) {
            throw new ObjectGridRuntimeException("Cannot push changes", e);
        }
    }
```

Este método utiliza diversas variáveis de instância:

- Variável `jmsSession`: Uma sessão JMS utilizada para publicar mensagens. É criada quando o plug-in inicializa.
- Variável `mode`: O modo de distribuição.
- Variável `publishMaps`: Um conjunto que contém o nome de cada mapa com alterações para publicar. Se a variável está vazia, então, todos os mapas são publicados.
- Variável `publisher`: Um objeto TopicPublisher que é criado durante o método de inicialização de plug-in

Receber e aplicar mensagens de atualização

A seguir está o método de execução. Este método é executado em um loop até que o aplicativo pare o loop. Cada iteração do loop tenta receber uma mensagem JMS e aplicá-la ao ObjectGrid.

Exemplo do método de execução de mensagem JMS

```
private synchronized boolean isListenerRunning() {
        return listenerRunning;
    }
```

```
public void run () {
    try {
        System.out.println("Listener starting");
        // get a jms session for receiving the messages.
        // Non transactional.
        TopicSession myTopicSession;
        myTopicSession = connection.createTopicSession(false, javax.jms.
   Session.AUTO_ACKNOWLEDGE);

        // get a subscriber for the topic, true indicates don't receive
        // messages transmitted using publishers
        // on this connection. Otherwise, we'd receive our own updates.
        TopicSubscriber subscriber = myTopicSession.createSubscriber(topic, null, true);
        System.out.println("Listener started");
        while (isListenerRunning()) {
            ObjectMessage om = (ObjectMessage)subscriber.receive(2000);
            if (om != null) {
                // Use Session that was passed in on the initialize...
                // very important to use no write through here
                mySession.beginNoWriteThrough();
                byte[] raw = (byte[])om.getObject();
                ByteArrayInputStream bis = new ByteArrayInputStream(raw);
                ObjectInputStream ois = new ObjectInputStream(bis);
                // inflate the LogSequences
                Collection collection = LogSequenceTransformer.inflate(ois, myGrid);
                Iterator iter = collection.iterator();
                while (iter.hasNext()) {
                    // process each Maps changes according to the mode when
                    // the LogSequence was serialized
                    LogSequence seq = (LogSequence)iter.next();
                    mySession.processLogSequence(seq);
                }
                mySession.commit();
            } // if there was a message
        } // loop while
        // stop the connection
        connection.close();
    } catch (IOException e) {
        System.out.println("IO Exception: " + e);
    } catch (JMSException e) {
        System.out.println("JMS Exception: " + e);
    } catch (ObjectGridException e) {
        System.out.println("ObjectGrid exception: " + e);
        System.out.println("Caused by: " + e.getCause());
    } catch (Throwable e) {
        System.out.println("Exception : " + e);
    }
    System.out.println("Listener stopped");
}
```

Listener de Eventos da JMS

O JMSObjectGridEventListener foi projetado para suportar invalidação de cache próximo do lado do cliente e um mecanismo de replicação ponto a ponto. Essa é uma implementação Java Message Service (JMS) da interface ObjectGridEventListener.

O mecanismo de invalidação do cliente pode ser usado em um ambiente distribuído doeXtreme Scale para garantir que os dados de cache próximos do cliente sejam sincronizados com os servidores ou outros cliente. Sem essa função, o cache perto do cliente pode deixar os dados obsoletos. Entretanto, mesmo com essa invalidação de cliente baseada em JMS, é necessário levar em consideração a janela de sincronização para atualizar um cache perto do cliente devido ao atraso do tempo de execução ao publicar as atualizações.

O mecanismo de replicação ponto a ponto pode ser usado em ambientes local e distribuído do eXtreme Scale. Ele é um processo de replicação de núcleo para núcleo do ObjectGrid e permite o fluxo de atualizações de dados entre os ObjectGrids locais e distribuídos. Por exemplo, com esse mecanismo, é possível mover as atualizações de dados de uma grade distribuída para um ObjectGrid local ou de qualquer grade para outra grade em um domínio de sistema diferente.

O JMSObjectGridEventListener requer que o usuário configure as informações do JMS e do Java Naming and Directory Interface (JNDI) para obter os recursos do

JMS necessários. Além disso, as propriedades relacionadas à replicação devem ser configuradas corretamente. Em um ambiente JEE, o JNDI deve estar disponível em ambos os contêineres da Web e do Enterprise JavaBean (EJB). Nesse caso, a propriedade JNDI é opcional, a menos que você deseje obter os recursos JMS externos.

Esse listener de evento possui propriedades que podem ser configuradas com o XML ou com abordagens programáticas, que podem ser usadas apenas para invalidação do cliente, apenas replicação ponto a ponto, ou ambos. A maioria das propriedades é opcional para customizar o comportamento para obter a funcionalidade necessária.

.

Para obter mais informações, consulte a API do JMSObjectGridEventListener.

Estendendo o Plug-in do JMSObjectGridEventListener

O plug-in do JMSObjectGridEventListener permite que as instâncias equivalentes do ObjectGrid recebam atualizações quando os dados na grade forem alterados ou despejados. Também permite que os clientes sejam notificados quando as entradas forem atualizadas ou despejadas de uma grade do eXtreme Scale. Esse tópico descreve como estender o plug-in do JMSObjectGridEventListener para permitir que os aplicativos sejam notificados quando uma mensagem JMS for recebida. Isso é mais útil ao usar a configuração do CLIENT_SERVER_MODEL para invalidação do cliente.

Ao executar na função do receptor, o método JMSObjectGridEventListener.onMessage substituído será chamado automaticamente pelo tempo de execução do eXtreme Scale quando a instância do JMSObjectGridEventListener receber atualizações de mensagem JMS da grade. Essas mensagens agrupam uma coleta de Objetos LogSequence. Os objetos LogSequence são transmitidos para o método onMessage e o aplicativo usa o LogSequence para identificar quais entradas de cache foram inseridas, excluídas, atualizadas ou invalidadas.

Para usar o ponto de extensão onMessage, os aplicativos executam as seguintes etapas.

1. Criar uma nova classe, estendendo a classe JMSObjectGridEventListener, substituindo o método onMessage.
2. Configurar o JMSObjectGridEventListener estendido da mesma forma que o ObjectGridEventListener para ObjectGrid.

O JMSObjectGridEventListener estendido é uma classe filha do JMSObjectGridEventListener e pode, opcionalmente, substituir dois métodos: inicializar (opcional) e onMessage. Se uma classe filha do JMSObjectGridEventListener precisar usar algum artefato do ObjectGrid, como um ObjectGrid ou de Sessão no método onMessage, ele poderá obter esses artefatos no método initialize e armazená-los em cache como variáveis da instância. Além disso, no método onMessage, os artefatos ObjectGrid em cache podem ser usados para processar uma coleta de LogSequences transmitida.

Nota: O método de inicialização substituído deve chamar o método super.initialize para inicializar o JMSObjectGridEventListener pai apropriadamente.

A seguir há uma amostra de uma classe JMSObjectGridEventListener estendida.

```
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;

    /**
     * Essa é a grade associada a esse listener.
     */
    ObjectGrid grid;

    /**
     * Essa é a sessão associada a esse listener.
     */
    Session session;

    String objectGridType;

    public List receivedLogSequenceList = new ArrayList();


 /* (não-Javadoc)
  * @see com.ibm.websphere.objectgrid.plugins.builtins.JMSObjectGridEventListener
    #initialize(com.ibm.websphere.objectgrid.Session)
  */
 public void initialize(Session session) {
  // Nota: se for necessário usar qualquer artefato ObjectGrid, essa classe precisará obter ObjectGrid
  //  a partir da instância de Sessão transmitida e obter o ObjectMap da instância de sessão
  // para qualquer operação de mapa ObjectGrid transacional.

  super.initialize(session);  // deve chamar o método initialize super.
  this.session = session; // armazenar em cache a instância da sessão, caso tiver que
  //   usá-lo para executar a operação de mapa.
  this.grid = session.getObjectGrid(); // get ObjectGrid, caso tiver que obter
  //   informações do ObjectGrid.

  if (grid.getObjectGridType() == ObjectGrid.CLIENT)
   objectGridType = "CLIENT";
  else if (grid.getObjectGridType() == ObjectGrid.SERVER)
   objectGridType = "Server";

  if (debug)
   System.out.println("ExtendedJMSObjectGridEventListener[" +
     objectGridType + "].initialize() : grid = " + this.grid);
 }

 /* (não-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 + "]";
 }
}
```

Configuração

A classe JMSObjectGridEventListener estendida deve ser configurada da mesma forma para o mecanismo de invalidação de cliente e de replicação ponto a ponto. A seguir há um exemplo de configuração XML.

```
<objectGrid name="PRICEGRID">
   <bean id="ObjectGridEventListener"
    className="com.ibm.websphere.samples.objectgrid.jms.
      price.ExtendedJMSObjectGridEventListener">
    <property name="invalidationModel" type="java.lang.String" value="CLIENT_SERVER_MODEL" description="" />
    <property name="invalidationStrategy" type="java.lang.String"
     value="INVALIDATE" description="" />
    <property name="jms_topicConnectionFactoryJndiName" type="java.lang.String"
     value="jms/TCF" description="" />
    <property name="jms_topicJndiName" type="java.lang.String"
     value="GRID.PRICEGRID" description="" />
    <property name="jms_topicName" type="java.lang.String"
     value="GRID.PRICEGRID" description="" />
    <property name="jms_userid" type="java.lang.String" value="" description="" />
    <property name="jms_password" type="java.lang.String" value="" description="" />
   </bean>
   <backingMap name="PRICE" pluginCollectionRef="PRICE"></backingMap>
      </objectGrid>
```

Nota: ClassName do bean ObjectGridEventListener é configurado com a classe JMSObjectGridEventListener estendida com as mesmas propriedades que o JMSObjectGridEventListener genérico.

Configurando com os Arquivos XML

O WebSphere eXtreme Scale é configurado por uma coleta de arquivos XML. Cada arquivo XML tem uma finalidade na configuração do produto.

Configuração de Topologia de Implementação

Use o XML descritor de política de implementação para gerenciar sua topologia de implementação.

A política de implementação é codificada como um arquivo XML que é fornecido para o contêiner do eXtreme Scale. O arquivo XML especifica as seguintes informações.

- Os mapas que pertencem a cada conjunto de mapas.
- O número de partições
- O número de réplicas síncronas e assíncronas

A política de implementação também controla os seguintes comportamentos de colocação.

- O número mínimo de contêineres ativos antes da ocorrência da disposição
- A substituição automática de shards perdidos
- A disposição de cada shard de uma única partição em uma máquina diferente.

Para obter mais informações sobre o arquivo XML de política de implementação, consulte “Arquivo Descritor XML de Política de Implementação” na página 145.

As informações de terminal não estão pré-configuradas no ambiente dinâmico. Não há nomes de servidor ou informações de topologia física localizadas na política de implementação. Todos os shards em uma grade dinâmica são automaticamente colocados em contêineres pelo serviço de catálogo. O serviço de catálogo usa as restrições que são definidas pela política de implementação para gerenciar automaticamente a colocação de shard. Essa colocação automática de shard facilita a configuração de grades maiores. Também é possível incluir servidores no seu ambiente, conforme necessário.

Nota: Em um ambiente do WebSphere Application Server, um tamanho de grupo principal de mais de 50 membros não é suportado.

Um arquivo XML de política de implementação é passado para um contêiner eXtreme Scale durante a inicialização. Uma política de implementação deve ser usada junto com o arquivo XML de ObjectGrid. A política de implementação não é requerida para iniciar um contêiner, mas é recomendada. A política de implementação deve ser compatível com o ObjectGrid XML utilizado com ela. Para cada elemento objectgridDeployment na política de implementação, você deve ter um elemento objectGrid correspondente no XML do seu ObjectGrid. Os mapas no objectgridDeployment devem ser consistentes com os backingMaps localizados no XML ObjectGrid. Cada backingMap deve ser referido dentro de um e apenas um mapSet.

No exemplo a seguir, o arquivo `companyGridDpReplication.xml` é destinado a formar um par com o arquivo `companyGrid.xml` correspondente.

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>
```

O arquivo `companyGridDpReplication.xml` possui um mapSet que é dividido em 11 partições. Cada partição deve ter exatamente uma réplica síncrona. O número de réplicas síncronas é dedicado pelos atributos minSyncReplicas e maxSyncReplicas. Como o atributo minSyncReplicas é configurado para 1, cada partição no mapSet deve ter pelo menos uma réplica síncrona disponível para processar transações de

gravação. Como maxSyncReplicas é configurado para 1, cada partição deve ter no máximo 1 réplica síncrona. As partições nesse mapSet não possuem réplicas.

O atributo numInitialContainers instrui o serviço de catálogo para adiar a colocação até que quatro contêineres estejam disponíveis para suportar esse ObjectGrid. O atributo numInitialContainers será ignorado depois que o número especificado de contêineres for alcançado.

O arquivo `companyGridDpReplication.xml` demonstra uma maneira comum de configurar uma política de implementação, mas uma política de implementação pode oferecer até mesmo mais controle sobre como e quando seu ObjectGrid é implementado em seu ambiente. Para obter uma descrição completa do arquivo descritor de política de implementação, consulte "Arquivo Descritor XML de Política de Implementação" na página 145.

Topologia Distribuída

Caches consistentes distribuídos oferecem melhores desempenho, disponibilidade e escalabilidade que podem ser configurados.

O WebSphere eXtreme Scale equilibra automaticamente os servidores. É possível incluir servidores adicionais sem reiniciar o WebSphere eXtreme Scale. Incluir servidores adicionais sem precisar reiniciar o eXtreme Scale permite executar implementações simples e implementações que atingem terabytes em que milhares de servidores são necessários. Essa topologia de implementação é flexível. Com o serviço de catálogo, é possível incluir e remover servidores para melhorar o uso de recursos sem remover o cache inteiro. Para incluir ou remover um servidor, use o comando startOgServer para iniciar um servidor de contêiner, que se comunica com o serviço de catálogo com a opção **-catalogServiceEndPoints**. Todos os clientes de topologia distribuída se comunicam com o serviço de catálogo através do Internet Interoperability Object Protocol (IIOP). Todos os clientes utilizam a interface ObjectGrid para comunicar-se com servidores.

O recurso de configuração dinâmica do WebSphere eXtreme Scale facilita incluir recursos no sistema. Contêineres hospedam os dados e o serviço de catálogo funciona como o ponto de acesso para a grade. Os contêineres são responsáveis pela manutenção dos dados. O serviço de catálogo é responsável por encaminhar pedidos para o local correto no primeiro acesso, alocando espaço em contêineres de host e gerenciando o funcionamento e a disponibilidade geral do sistema. Os clientes se conectam a um serviço de catálogo, recuperam uma descrição da topologia de contêiner-servidor e, em seguida, comunica-se diretamente a cada servidor, conforme necessário. Quando a topologia do servidor é alterada porque novos servidores foram incluídos, ou em razão de falha de outros, o cliente é automaticamente redirecionado para o servidor apropriado que está hospedando os dados.

Um serviço de catálogo normalmente existe na sua própria grade do Java Virtual Machines. Um serviço de catálogo único pode ser utilizado para gerenciar vários servidores. Um contêiner pode ser iniciado em uma JVM por si ou pode ser carregado em uma JVM arbitrária com outros contêineres para diferentes servidores. Pode existir um cliente em alguma JVM e comunicar-se com um ou mais servidores. Um cliente também pode existir na mesma JVM como um contêiner.

Arquivo Descritor XML de Política de Implementação

Para configurar uma política de implementação, utilize um arquivo XML do descritor da política de implementação.

Nas seguintes seções, os elementos e atributos do arquivo XML do descritor da política de implementação está definido. Consulte o "Arquivo deploymentPolicy.xsd" na página 148 para obter um exemplo do esquema XML de política de implementação.

Elemento deploymentPolicy

O elemento deploymentPolicy é o elemento de nível superior do arquivo XML de política de implementação. Este elemento configura o espaço de nomes do arquivo e o local do esquema. O esquema está definido no arquivo `deploymentPolicy.xsd`.

- Número de ocorrências: Uma
- Elemento-filho: Elemento objectgridDeployment

Elemento objectgridDeployment

O elemento objectgridDeployment é utilizado para referenciar uma instância do ObjectGrid a partir do arquivo XML do ObjectGrid. Dentro do elemento objectgridDeployment, é possível dividir seus mapas em conjuntos de mapas.

- Número de ocorrências: Uma ou mais
- Elemento-filho: Elemento mapSet

Atributos

objectgridName
Especifica o nome do ObjectGrid para implementar. Esse atributo faz referência a um elemento objectGrid que está definido no arquivo XML do ObjectGrid. (Necessário)

```
<objectgridDeployment objectgridName="objectgridName"/>
```

Por exemplo, o atributo objectgridName é configurado como `CompanyGrid` no arquivo `companyGridDpReplication.xml`. O atributo objectgridName referencia o `CompanyGrid` que é definido no arquivo `companyGrid.xml`. Para obter mais informações, consulte "Arquivo XML descritor do ObjectGrid" na página 150.

Elemento mapSet

O elemento mapSet é utilizado para agrupar mapas. Os mapas dentro de um elemento mapSet são particionados e replicados da mesma maneira. Cada mapa deve pertencer a apenas um mapSet.

- Número de ocorrências: Uma ou mais
- Elemento-filho: Elemento map

Atributos

name
Especifica o nome do mapSet. Esse atributo deve ser exclusivo dentro do elemento objectgridDeployment. (Necessário)

numberOfPartitions
Especifica o número de partições para o mapSet. O padrão é 1. O número deve ser apropriado para o número de contêineres que hospedam as partições. (Opcional)

minSyncReplicas
Especifica o número mínimo de réplicas assíncronas para cada partição no mapSet. O padrão é 0. Shards não são colocados até que o domínio possa suportar o número mínimo de réplicas síncronas. Para suporte do valor minSyncReplicas, é preciso de mais um contêiner do que o valor de minSyncReplicas. Se o número de réplicas síncronas ficar abaixo do valor de minSyncReplicas, grave transações que não são mais permitidas àquela partição. (Opcional)

maxSyncReplicas
Especifica o número máximo de réplicas síncrona para cada partição no mapSet. O padrão é 0. Nenhuma outra réplica síncrona é colocada para uma partição após um domínio alcançar este número de réplicas síncronas para esta partição específica. A inclusão de contêineres que podem suportar esse ObjectGrid pode resultar em um aumento no número de réplicas síncronas se seu valor de maxSyncReplicas ainda não tiver sido atingido. (Opcional)

maxAsyncReplicas
Especifica o número máximo de réplicas assíncronas para cada partição no mapSet. O padrão é 0. Após o primário e todas as réplicas síncronas terem sido colocadas para uma partição, as réplicas assíncronas são colocadas até que o valor maxAsyncReplicas seja correspondido. (Opcional)

replicaReadEnabled
Se este atributo é configurado como `true`, pedidos de leitura são distribuídos entre uma partição primária e suas réplicas. Se o atributo replicaReadEnabled for `false`, os pedidos de leitura são roteados para o primário apenas. O padrão é false. (Opcional)

numInitialContainers
Especifica o número de contêineres do eXtreme Scale que são necessários antes que ocorra a disposição inicial para os shards neste mapSet. O padrão é 1. Este atributo pode ajudar a economizar CPU e largura de banda ao tornar um ObjectGrid on-line. (Opcional)

Iniciar um contêiner do eXtreme Scale envia um evento para o serviço de catálogo. Na primeira vez em que o número de contêineres ativos for igual ao valor numInitialContainers para um mapSet, o serviço de catálogo colocará os shards a partir do mapSet, contanto que o minSyncReplicas também possa ser satisfeito. Depois que o valor numInitialContainers tiver sido atendido, cada contêiner que iniciou um evento pode acionar um reequilíbrio de shards não colocados e colocados anteriormente. Se você souber aproximadamente quantos contêineres estará iniciando para esse mapSet, poderá configurar o valor de numInitialContainers perto desse número para evitar o reequilíbrio após cada início de contêiner. A disposição não ocorre até que você atinja o valor de numInitialContainers para o mapSet.

autoReplaceLostShards
Especifica se os shards perdidos são colocados em outros contêineres. O padrão é true. Quando um contêiner é parado ou falha, os shards executando no contêiner são perdidos. Um primário perdido tem uma de suas réplicas promovida para ser o novo primário. Devido a essa promoção, uma das réplicas é perdida. Em determinados casos, você pode não querer que seus shards perdidos sejam automaticamente substituídos nos contêineres disponíveis. Se desejar que os shards perdidos permaneçam não-colocados, configure o atributo autoReplaceLostShards como `false`. Essa configuração não afeta a cadeia de promoção, mas somente a substituição do último shard na cadeia. (Opcional)

developmentMode
Com este atributo, é possível influenciar onde um shard é colocado em relação a seus shards peer. O padrão é true. Quando o atributo developmentMode é configurado para `false`, nenhum dos dois shards da mesma partição é colocado na mesma máquina. Quando o atributo developmentMode é configurado para `true`, os shards da mesma partição podem ser colocados na mesma máquina. Nos dois casos, nenhum dos dois shards da mesma partição são nem mesmo colocados no mesmo contêiner. (Opcional)

placementStrategy
Há duas estratégias de colocação. A estratégia padrão é `FIXED_PARTITION`, em que o número de shards primários que são colocados nos contêineres disponíveis é igual ao número de partições definidas, aumentado pelo número de réplicas. A estratégia alternativa é `PER_CONTAINER`, em que o número de shards primários que são colocados em cada contêiner é igual ao número de partições definidas, com um número igual de réplicas colocadas em outros contêineres. (Opcional)

```
<mapSet
(1)   name="mapSetName"
(2)   numberOfPartitions="numberOfPartitions"
(3)   minSyncReplicas="minimumNumber"
(4)    maxSyncReplicas="maximumNumber"
(5)   maxAsyncReplicas="maximumNumber"
(6)   replicaReadEnable="true" | "false"
(7)   numInitialContainers="numberOfInitialContainersBeforePlacement"
(8)   autoReplaceLostShards="true" | "false"
(9)   developmentMode="true" | "false"
(10)   placementStrategy="FIXED_PARTITION"|"PER_CONTAINER"
/>
```

No exemplo a seguir, o elemento mapSet é utilizado para configurar uma política de implementação. O valor é configurado como `mapSet1` e é dividido em 10 partições. Cada uma dessas partições tem pelo menos uma réplica síncrona disponíveis e não mais do que duas réplicas síncronas. Cada partição também tem uma réplica assíncrona se o ambiente a suportar. Todas as réplicas síncronas são colocadas antes que quaisquer réplicas assíncronas sejam colocadas. Além disso, o serviço de catálogo não tenta colocar os shards do mapSet1 até que o domínio possa suportar o valor de minSyncReplicas. O suporte do valor minSyncReplicas requer dois ou mais contêineres: um para o primário e dois para a réplica síncrona.

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

 <objectgridDeployment objectgridName="CompanyGrid">
  <mapSet name="mapSet1" numberOfPartitions="10"
   minSyncReplicas="1" maxSyncReplicas="2" maxAsyncReplicas="1"
    numInitialContainers="10" autoReplaceLostShards="true"
   developmentMode="false" replicaReadEnabled="true">
   <map ref="Customer" />
   <map ref="Item" />
   <map ref="OrderLine" />
   <map ref="Order" />
  </mapSet>
 </objectgridDeployment>

</deploymentPolicy>
```

Mesmo que apenas dois contêineres sejam necessários para atender às configurações de replicação, o atributo numInitialContainers requer 10 contêineres disponíveis antes que o serviço de catálogo tente colocar qualquer um dos shards neste mapSet. Depois que o domínio tiver 10 contêineres que podem suportar o CompanyGrid ObjectGrid, todos os shards no mapSet1 são colocados.

Como o atributo autoReplaceLostShards está configurado para true, qualquer shard nesse mapSet que for perdido como resultado de falha no contêiner é automaticamente substituído por outro contêiner, desde que esse contêiner esteja

disponível para hospedar o shard perdido. Shards da mesma partição não podem ser colocados na mesma máquina para mapSet1 porque o atributo developmentMode é configurado como `false`. Pedidos de leitura são distribuídos através do primário e suas réplicas de cada partição porque o valor de replicaReadEnabled está `true`.

Elemento map

Cada mapa em um elemento mapSet refere-se a um dos elementos backingMap definidos no arquivo XML do ObjectGrid. Cada mapa em um eXtreme Scale distribuído deve pertencer a apenas um elemento mapSet. Consulte o "Arquivo `objectGrid.xsd`" na página 166 para obter mais informações sobre o arquivo XML ObjectGrid.

- Número de ocorrências: Uma ou mais
- Elemento-filho: Nenhum

Atributos

ref

Fornece uma referência para um elemento backingMap no arquivo XML do ObjectGrid. Cada mapa em um mapSet deve fazer referência a um backingMap a partir do arquivo XML do ObjectGrid. O valor que está atribuído para o atributo ref deve corresponder ao atributo de nome de um dos elementos backingMap no arquivo XML do ObjectGrid. (Necessário)

```
<map
(1)   ref="backingMapReference"
/>
```

O arquivo `companyGridDpMapSetAttr.xml` utiliza o atributo ref no mapa para fazer referência a cada um dos backingMaps do arquivo `companyGrid.xml`.

Para obter informações sobre como evitar conflitos de configuração XML, consulte "Resolução de Problemas de Configuração XML" na página 81.

Arquivo `deploymentPolicy.xsd`:

Utilize o esquema XML de política de implementação para criar um arquivo XML descritor de implementação.

Consulte o "Arquivo Descritor XML de Política de Implementação" na página 145 para obter descrições dos elementos e atributos definidos no arquivo `deploymentPolicy.xsd`.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:dp="http://www.ibm.com/ws/objectgrid/deploymentPolicy"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNameSpace="http://ibm.com/ws/objectgrid/deploymentPolicy"
 elementFormDefault="qualified">

 <xsd:element name="deploymentPolicy">
  <xsd:complexType>
   <xsd:choice>
    <xsd:element name="objectgridDeployment"
     type="dp:objectgridDeployment" minOccurs="1"
     maxOccurs="unbounded">
     <xsd:unique name="mapSetNameUnique">
      <xsd:selector xpath="dp:mapset" />
      <xsd:field xpath="@name" />
     </xsd:unique>
    </xsd:element>
   </xsd:choice>
  </xsd:complexType>
 </xsd:element>

 <xsd:complexType name="objectgridDeployment">
  <xsd:sequence>
   <xsd:element name="mapSet" type="dp:mapSet"
```

```
   maxOccurs="unbounded" minOccurs="1">
   <xsd:unique name="mapNameUnique">
    <xsd:selector xpath="dp:map" />
    <xsd:field xpath="@ref" />
   </xsd:unique>
  </xsd:element>
 </xsd:sequence>
 <xsd:attribute name="objectgridName" type="xsd:string"
  use="required" />
</xsd:complexType>

<xsd:complexType name="mapSet">
 <xsd:sequence>
  <xsd:element name="map" type="dp:map" maxOccurs="unbounded"
   minOccurs="1" />
  <xsd:element name="zoneMetadata" type="dp:zoneMetadata"
   maxOccurs="1" minOccurs="0">

   <xsd: key name="zoneRuleName">
    <xsd:selector xpath="dp:zoneRule" />
    <xsd:field xpath="@name" />
   </xsd:key>

   <xsd:keyref name="zoneRuleRef"
    refer="dp:zoneRuleName">
    <xsd:selector xpath="dp:shardMapping" />
    <xsd:field xpath="@zoneRuleRef" />
   </xsd:keyref>

  </xsd:element>
 </xsd:sequence>
 <xsd:attribute name=="name" type="xsd:string" use="required" />
 <xsd:attribute name="numberOfPartitions" type="xsd:int"
  use="optional" />
 <xsd:attribute name="minSyncReplicas" type="xsd:int"
  use="optional" />
 <xsd:attribute name="maxSyncReplicas" type="xsd:int"
  use="optional" />
 <xsd:attribute name="maxAsyncReplicas" type="xsd:int"
  use="optional" />
 <xsd:attribute name="replicaReadEnabled" type="xsd:boolean"
  use="optional" />
 <xsd:attribute name="numInitialContainers" type="xsd:int"
  use="optional" />
 <xsd:attribute name="autoReplaceLostShards" type="xsd:boolean"
  use="optional" />
 <xsd:attribute name="developmentMode" type="xsd:boolean"
  use="optional" />
 <xsd:attribute name="placementStrategy"
  type="dp:placementStrategy" use="optional" />
</xsd:complexType>

<xsd:simpleType name="placementStrategy">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="FIXED_PARTITIONS" />
  <xsd:enumeration value="PER_CONTAINER" />
 </xsd:restriction>
</xsd:simpleType>


<xsd:complexType name="map">
 <xsd:attribute name="ref" use="required" />
</xsd:complexType>

<xsd:complexType name="zoneMetadata">
 <xsd:sequence>
  <xsd:element name="shardMapping" type="dp:shardMapping"
   maxOccurs="unbounded" minOccurs="1" />
  <xsd:element name="zoneRule" type="dp:zoneRule"
   maxOccurs="unbounded" minOccurs="1">

  </xsd:element>

 </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="shardMapping">
 <xsd:attribute name="shard" use="required">
  <xsd:simpleType>
   <xsd:restriction base="xsd:string">
    <xsd:enumeration value="P"></xsd:enumeration>
    <xsd:enumeration value="S"></xsd:enumeration>
    <xsd:enumeration value="A"></xsd:enumeration>
   </xsd:restriction>
  </xsd:simpleType>
 </xsd:attribute>
 <xsd:attribute name="zoneRuleRef" type="xsd:string"
  use="required" />
</xsd:complexType>

<xsd:complexType name="zoneRule">
```

```
  <xsd:sequence>
   <xsd:element name="zone" type="dp:zone"
    maxOccurs="unbounded" minOccurs="1" />
  </xsd:sequence>
  <xsd:attribute name="name" type="xsd:string" use="required" />
  <xsd:attribute name="exclusivePlacement" type="xsd:boolean" />
   use="optional" />
 </xsd:complexType>

 <xsd:complexType name="zone">
  <xsd:attribute name="name" type="xsd:string" use="required" />
 <xsd:complextType>

</xsd:schema>
```

Arquivo XML descritor do ObjectGrid

Para configurar o WebSphere eXtreme Scale, utilize um arquivo XML descritor do ObjectGrid e a API do ObjectGrid.

Nas seções a seguir, os arquivos XML de amostra são fornecidos para ilustrar diversas configurações. Cada elemento e atributo do arquivo XML está definido. Use o esquema de descritor XML do ObjectGrid para criar o arquivo descritor XML. Consulte o "Arquivo objectGrid.xsd" na página 166 para obter um exemplo do esquema do descritor XML do ObjectGrid.

Uma versão modificada do arquivo companyGrid.xml original é utilizada. O arquivo companyGridSingleMap.xml a seguir é semelhante ao arquivo companyGrid.xml. O arquivo companyGridSingleMap.xml tem um mapa e o arquivo companyGrid.xml tem quatro mapas. Os elementos e os atributos do arquivo são descritos em detalhes no exemplo a seguir.

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

  <objectGrids>
    <objectGrid name="CompanyGrid">
      <backingMap name="Customer"/>
    </objectGrid>
  </objectGrids>
</objectGridConfig>
```

Elemento objectGridConfig

O elemento objectGridConfig é o elemento de nível superior do arquivo XML. Grave este elemento em seu documento XML do eXtreme Scale, conforme mostrado no exemplo anterior. Este elemento configura o espaço de nomes do arquivo e o local do esquema. O esquema está definido no arquivo objectGrid.xsd.

- Número de ocorrências: Uma
- Elemento-filho: Elemento objectGrids e elemento backingMapPluginCollections

Elemento objectGrids

O elemento objectGrids é um contêiner para todos os elementos objectGrid. No arquivo companyGridSingleMap.xml, o elemento objectGrids contém um objectGrid, o CompanyGrid objectGrid.

- Número de ocorrências: Uma ou mais
- Elemento-filho: Elemento objectGrid

Elemento objectGrid

Utilize o elemento objectGrid para definir um ObjectGrid. Cada um dos atributos no elemento objectGrid corresponde a um método na interface ObjectGrid.

- Número de ocorrências: Uma para muitas
- Elemento-filho: Elemento bean, elemento backingMap, elemento querySchema e elemento streamQuerySet

Atributos

name
Especifica o nome que é designado ao ObjectGrid. A validação de XML falhará se esse atributo estiver faltando. (Necessário)

securityEnabled
Ativa a segurança no nível do ObjectGrid, o que possibilita as autorizações de acesso aos dados no mapa, quando você configura o atributo como `true`. O padrão é `true`. (Opcional)

authorizationMechanism
Configura o mecanismo de autorização para o elemento. É possível configurar o atributo com um dos dois valores: `AUTHORIZATION_MECHANISM_JAAS` ou `AUTHORIZATION_MECHANISM_CUSTOM`. O padrão é `AUTHORIZATION_MECHANISM_JAAS`. Configurado como `AUTHORIZATION_MECHANISM_CUSTOM` quando você estiver utilizando um plug-in MapAuthorization customizado. É necessário configurar o atributo securityEnabled como `true` para o atributo authorizationMechanism tomar efeito. (Opcional)

permissionCheckPeriod
Especifica um valor de número inteiro em segundos que indica com que frequência verificar a permissão que é utilizada para permitir um acesso do cliente. O padrão é `0`. Ao configurar o valor de atributo `0`, cada chamada de método get, put, update, remove ou evict solicita que o mecanismo de autorização, seja a autorização Java Authentication and Authorization Service (JAAS) ou a autorização customizada, verifique se o usuário atual possui permissão. Um valor maior do que 0 indica o número de segundos para armazenar em cache um conjunto de permissões antes de retornar ao mecanismo de autorização para atualizar. É necessário configurar o atributo securityEnabled como `true` para o atributo permissionCheckPeriod tomar efeito. (Opcional)

txTimeout
Especifica a quantidade de tempo em segundos permitida para que uma transação seja concluída. Se uma transação não for concluída neste período de tempo, será marcada para reversão e como resultado, ocorrerá uma exceção TransactionTimeoutException. Se você configurar o valor como `0`, as transações nunca expirarão. (Opcional)

entityMetadataXMLFile
Especifica o caminho relativo para o arquivo XML descritor da entidade. O caminho é referente ao local do arquivo descritor do Objectgrid. Utilize este atributo para definir um esquema de entidade utilizando um arquivo XML. As entidades devem ser definidas antes de iniciar o eXtreme Scale de forma que cada entidade possa vincular-se a um BackingMap. (Opcional)

```
<objectGrid
(1)   name="objectGridName"
(2)   securityEnabled="true" | "false"
(3)   authorizationMechanism="AUTHORIZATION_MECHANISM_JASS" | "AUTHORIZATION_MECHANISM_CUSTOM"
```

```
(4)    permissionCheckPeriod="permission_check_period"
(5)   txTimeout="seconds"
(6)   entityMetadataXMLFile="URL"
/>
```

No exemplo a seguir, o arquivo `companyGridObjectGridAttr.xml` demonstra uma maneira de configurar os atributos de um elemento objectGrid. A segurança é ativada, o mecanismo de autorização é configurado como JAAS e o período de verificação da permissão é configurado para 45 segundos. O arquivo também registra entidades especificando um atributo entityMetadataXMLFile.

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

 <objectGrids>
  <objectGrid name="CompanyGrid" securityEnabled="true"
   authorizationMechanism="AUTHORIZATION_MECHANISM_JASS"
   permissionCheckPeriod="45"
   entityMetadataXMLFile="companyGridEntities.xml">
   <backingMap name="Customer"/>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

A amostra de código a seguir demonstra a abordagem programática para obter a mesma configuração que o arquivo `companyGridObjectGridAttr.xml` no exemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);

companyGrid.setSecurityEnabled();
companyGrid.setAuthorizationMechanism(SecurityConstants.AUTHORIZATION_MECHANISM_JAAS);
companyGrid.setPermissionCheckPeriod(45);
companyGrid.registerEntities(new URL("file:companyGridEntities.xml"));
```

Elemento backingMap

O elemento backingMap é utilizado para definir uma instância BackingMap em um ObjectGrid. Cada um dos atributos no elemento backingMap corresponde a um método na interface BackingMap. Para obter detalhes, consulte "Interface BackingMap" na página 63.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Elemento timeBasedDBUpdate

Atributos

name
Especifica o nome que é designado à instância do backingMap. Se este atributo estiver ausente, a validação de XML falhará. (Necessário)

readOnly
Configura uma instância do BackingMap como leitura-gravação quando você especifica o atributo como `false`. Quando você especifica o atributo como `true`, a instância do BackingMap é somente leitura. As chamadas de Session.getMap(String) resultarão na criação de mapa dinâmico se o nome transmitido para o método corresponder à expressão regular especificada no atributo de nome desse backingMap. O valor padrão é `false`. (Opcional)

template
Especifica se os mapas dinâmicos podem ser usados. Configure esse valor para `true` se o mapa BackingMap for um mapa de modelo. Os mapas de modelo podem ser usados para criar dinamicamente mapas, mesmo depois que o ObjectGrid ser inicializado. As chamadas de Session.getMap(String) resultarão

na criação de mapas dinâmicos se o nome transmitido para o método corresponder à expressão regular especificada no atributo de nome desse backingMap. O valor padrão é `false`. (Opcional)

pluginCollectionRef
Especifica uma referência para um plug-in do backingMapPluginCollection. O valor desse atributo deve corresponder ao atributo ID de um plug-in backingMapCollection. A validação falhará se não existir nenhum ID correspondente. Configure o atributo para reutilizar plug-ins do BackingMap. (Opcional)

numberOfBuckets
Especifica o número de depósitos para a instância do BackingMap utilizar. A instância do BackingMap utiliza um mapa hash para implementação. Se existirem muitas entradas no BackingMap, mais depósitos resultarão em melhor desempenho, porque o risco de colisões é menor à medida que o número de depósitos aumenta. Mais depósitos também resultam em maior simultaneidade. Especifique um valor de `0` para desativar o cache local em um cliente ao comunicar-se remotamente com o eXtreme Scale. Ao configurar o valor como `0` para um cliente, configure o valor apenas no arquivo descritor XML do ObjectGrid da substituição do cliente. (Opcional)

preloadMode
Configura o modo preload se um plug-in do utilitário de carga for configurado para esta instância do BackingMap. O valor padrão é `false`. Se o atributo estiver configurado como `true`, o método Loader.preloadMap(Session, BackingMap) será chamado de maneira assíncrona. Caso contrário, a execução do método é bloqueada ao carregar dados de forma que o cache fique indisponível até que o pré-carregamento seja concluído. O pré-carregamento ocorre durante a inicialização. (Opcional)

lockStrategy
Especifica se o gerenciador de bloqueio interno é utilizado sempre que uma entrada de mapa é acessada por uma transação. Configure esse atributo com um dos três valores: `OPTIMISTIC`, `PESSIMISTIC` ou `NONE`. O valor padrão é `OPTIMISTIC`. (Opcional)

A estratégia de bloqueio otimista é normalmente utilizada quando um mapa que não possui um plug-in do utilitário de carga for mais lido do que gravado ou atualizado e quando o bloqueio não for fornecido pelo gerenciador de persistência usando o eXtreme Scale como um cache secundário ou pelo aplicativo. Um bloqueio exclusivo é obtido em uma entrada do mapa que é inserido, atualizado ou removido no momento do commit. O bloqueio assegura que as informações de versão não poderão ser alteradas por outra transação enquanto a transação que está sendo confirmada estiver executando uma verificação de versão otimista.

A estratégia de bloqueio pessimista é normalmente utilizada para um mapa que não possui um plug-in do utilitário de carga, e o bloqueio não é fornecido pelo gerenciador de persistência utilizando o eXtreme Scale como um cache secundário ou pelo aplicativo. A estratégia de bloqueio pessimista é utilizada quando a estratégia de bloqueio otimista falha com muita frequência porque as transações de atualização frequentemente colidem na mesma entrada do mapa.

A estratégia de ausência de bloqueio indica que o LockManager interno não é necessário. O controle de simultaneidade é fornecido fora do 3eXtreme Scale, seja pelo gerenciador de persistência usando o eXtreme Scale como um cache ou aplicativo secundário ou pelo plug-in do utilitário de carga que usa os bloqueios do banco de dados para controlar a simultaneidade.

Para obter informações adicionais, consulte “Bloqueio de Entrada de Mapa” na página 67.

numberOfLockBuckets
Configura o número de depósitos de bloquei que são utilizados pelo gerenciador de bloqueios para a instância do BackingMap. Configure o atributo lockStrategy como `OPTIMISTIC` ou `PESSIMISTIC` para criar um gerenciador de bloqueios para a instância do BackingMap. O gerenciador de bloqueios utiliza um mapa hash para rastrear as entradas que estão bloqueadas por uma ou mais transações. Se existirem muitas entradas, mais depósitos resultarão em melhor desempenho, porque o risco de colisões é menor à medida que o número de depósitos aumenta. Mais depósitos de bloqueios também resultam em maior simultaneidade. Configure o atributo lockStrategy como `NONE` para especificar que a instância do BackingMap não utilize nenhum gerenciador de bloqueios. (Opcional)

lockTimeout
Configura o tempo limite do bloqueio que é utilizado pelo gerenciador de bloqueios para a instância do BackingMap. Configure o atributo lockStrategy como `OPTIMISTIC` ou `PESSIMISTIC` para criar um gerenciador de bloqueios para a instância do BackingMap. Para evitar a ocorrência de conflitos, o gerenciador de bloqueios possui um valor de tempo limite de 15 segundos. Se o tempo limite for excedido, ocorre uma exceção LockTimeoutException. O valor padrão de 15 segundos é suficiente para a maioria dos aplicativos, mas em um sistema com grande extremamente carregado, um tempo limite pode ocorrer quando não existir nenhum conflito. Utilize o atributo lockTimeout para aumentar o valor do padrão para evitar a ocorrência de falsas exceções de tempo limite. Configure o atributo lockStrategy como `NONE` para especificar que a instância do BackingMap não utilize nenhum gerenciador de bloqueios. (Opcional)

CopyMode
Especifica se uma operação get de uma entrada na instância BackingMap retorna o valor real, uma cópia do valor ou um proxy para o valor. Configure o atributo CopyMode para um dos cinco valores:

`COPY_ON_READ_AND_COMMIT`
O valor padrão é `COPY_ON_READ_AND_COMMIT`. Configure o valor como `COPY_ON_READ_AND_COMMIT` para certificar-se de que um aplicativo nunca tenha uma referência para o objeto de valor que está na instância do BackingMap. Em vez disso, o aplicativo trabalha sempre com uma cópia do valor que está na instância de BackingMap. (Opcional)

`COPY_ON_READ`
Configure o valor como `COPY_ON_READ` para melhorar o desempenho sobre o valor `COPY_ON_READ_AND_COMMIT` eliminando a cópia que ocorre quando uma transação é confirmada. Para preservar a integridade dos dados do BackingMap, o aplicativo é confirmado para excluir cada referência para uma entrada após a transação ser confirmada. A configuração deste valor resulta em um método ObjectMap.get retornando uma cópia do valor ao invés de uma referência ao valor, o que garante que as alterações que são feitas pelo aplicativo no valor não afetem o elemento BackingMap até que a transação seja confirmada.

`COPY_ON_WRITE`
Configure o valor como `COPY_ON_WRITE` para melhorar o desempenho sobre o valor `COPY_ON_READ_AND_COMMIT` eliminando a cópia que ocorre quando o método ObjectMap.get é chamado pela primeira vez por uma transação para uma determinada chave. Em vez disso, o método

ObjectMap.get retorna um proxy para o valor em vez de uma referência direta ao objeto de valor. O proxy assegura que não seja feita uma cópia do valor, a menos que o aplicativo chame um método set na interface do valor.

NO_COPY
Configure o valor como NO_COPY para permitir que um aplicativo nunca modifique um objeto de valor que é obtido utilizando um método ObjectMap.get na troca de aprimoramentos de desempenho. Configure o valor como NO_COPY para mapas associados com as entidades da API do EntityManager.

COPY_TO_BYTES
Configure o valor para COPY_TO_BYTES para melhorar a área de cobertura da memória para tipos complexos do Object e para melhorar o desempenho quando a cópia de um Object depender da serialização para ser feita. Se um Object não for Clonável ou um ObjectTransformer com um método copyValue eficiente não for fornecido, o mecanismo de cópia padrão deverá serializar e aumentar o objeto para fazer uma cópia. Com a configuração COPY_TO_BYTES, o aumento é executado apenas durante a leitura e a serialização é executada apenas durante a confirmação.

Para obter mais informações sobre essas configurações, consulte as informações sobre as boas práticas do CopyMode no *Guia de Programação*..

valueInterfaceClassName
Especifica uma classe que é necessária ao configurar o atributo CopyMode como COPY_ON_WRITE. Esse atributo é ignorado para todos os outros modos. O valor COPY_ON_WRITE utiliza um proxy quando são feitas chamadas do método ObjectMap.get. O proxy assegura que não seja feita uma cópia do valor, a menos que o aplicativo chame um método set na classe especificada como o atributo valueInterfaceClassName. (Opcional)

copyKey
Especifica se uma cópia da chave é necessária quando uma entrada de mapa é criada. A cópia do objeto de chave permite que o aplicativo utilize o mesmo objeto de chave para cada operação de ObjectMap. Configure o valor como true para copiar o objeto de chave quando uma entrada de mapa é criada. O valor padrão é false. (Opcional)

nullValuesSupported
Configure o valor como true para suportar valores nulos no ObjectMap. Quando valores nulos são suportados, uma operação get que retorna null pode significar que o valor é nulo ou que o mapa não contém a chave que é passada para o método. O valor padrão é true. (Opcional)

ttlEvictorType
Especifica como o prazo de expiração de uma entrada do BackingMap é computado. Configure esse atributo com um dos três valores: CREATION_TIME, LAST_ACCESS_TIME ou NONE. O valor CREATION_TIME indica que um prazo de expiração de entrada é a soma do hora da criação da entrada mais o valor de atributo timeToLive. O valor LAST_ACCESS_TIME indica que um prazo de expiração da entrada é a soma da hora do último acesso da entrada mais o valor de atributo timeToLive. O valor NONE, que é o padrão, indica que uma entrada não possui prazo de expiração e está presente na instância do BackingMap até que o aplicativo explicitamente remova ou invalide a entrada. (Opcional)

timeToLive
Especifica, em segundos, quanto tempo cada entrada do mapa fica presente. O valor padrão de `0` significa que a entrada do mapa está presente para sempre, ou até que o aplicativo explicitamente remova ou invalide a entrada. Caso contrário, o evictor TTL despeja a entrada do mapa com base neste valor. (Opcional)

streamRef
Especifica que o backingMap é um mapa de origem do fluxo. Qualquer inserção ou atualização no backingMap é convertida em um evento de fluxo para o mecanismo de consulta de fluxo. Esse atributo deve fazer referência a um nome do fluxo válido em um streamQuerySet. (Opcional)

viewRef
Especifica que o backingMap é um mapa de visualização. A saída da visualização do mecanismo de consulta de fluxo é convertida no formato de tupla do eXtreme Scale e colocada no mapa. (Opcional)

evictionTriggers
Configura os tipos de acionadores de despejo adicionais a utilizar. Todos os evictors para o mapa de apoio utilizam esta lista de acionadores adicionais. Para evitar uma IllegalStateException, este atributo deve ser chamado antes do método ObjectGrid.initialize(). Além disso, observe que o método ObjectGrid.getSession() chama implicitamente o método ObjectGrid.initialize() se o método já foi chamado pelo aplicativo. As entradas na lista de acionadores são separadas por ponto-e-vírgulas. Os Current Eviction Triggers incluem MEMORY_USAGE_THRESHOLD. (Opcional)

```
<backingMap
(1)   name="objectGridName"
(2)   readOnly="true" | "false"
(3)      template="true" | "false"
(4)   pluginCollectionRef="reference to backingMapPluginCollection"
(5)   numberOfBuckets="number of buckets"
(6)   preloadMode="true" | "false"
(7)   lockStrategy="OPTIMISTIC" | "PESSIMISTIC" | "NONE"
(8)   numberOfLockBuckets="number of lock buckets"
(9)   lockTimeout="lock timeout"
(10)   copyMode="COPY_ON_READ_AND_COMMIT" | "COPY_ON_READ" | "COPY_ON_WRITE"
       | "NO_COPY" | "COPY_TO_BYTES"
(11)   valueInterfaceClassName="value interface class name"
(12)   copyKey="true" | "false"
(13)   nullValuesSupported="true" | "false"
(14)   ttlEvictorType="CREATION_TIME" | "LAST_ACCESS_TIME|NONE"
(15)   timeToLive="time to live"
(16)   streamRef="reference to a stream"
(17)   viewRef="reference to a view"
(18)   writeBehind="write-behind parameters"
(19)   evictionTriggers="MEMORY_USAGE_THRESHOLD"
/>
```

No exemplo a seguir, o arquivo `companyGridBackingMapAttr.xml` é utilizado para demonstrar uma configuração do backingMap de amostra.

```
<?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>
```

O código de amostra a seguir demonstra a abordagem programática para a obtenção da mesma configuração que o arquivo `companyGridBackingMapAttr.xml` no exemplo anterior:

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);

BackingMap customerMap = companyGrid.defineMap("Customer");
customerMap.setReadOnly(true);
customerMap.setNumberOfBuckets(641);
customerMap.setPreloadMode(false);
customerMap.setLockStrategy(LockStrategy.OPTIMISTIC);
customerMap.setNumberOfLockBuckets(409);
customerMap.setLockTimeout(30);

// ao configurar o modo de cópia como COPY_ON_WRITE, é requerida uma classe valueInterface
customerMap.setCopyMode(CopyMode.COPY_ON_WRITE,
 com.ibm.websphere.samples.objectgrid.CounterValueInterface.class);
customerMap.setCopyKey(true);
customerMap.setNullValuesSupported(false);
customerMap.setTtlEvictorType(TTLType.LAST_ACCESS_TIME);
customerMap.setTimeToLive(3000); // set time to live to 50 minutes
```

Elemento bean

Utilize o elemento bean para definir plug-ins. É possível conectar plug-ins nas instâncias dos elementos objectGrid e BackingMap.

- Número de ocorrências no elemento objectGrid: Zero para muitas
- Número de ocorrências no elemento backingMapPluginCollection: Zero para muitas
- Elemento-filho: Elemento property

Atributos

id Especifica o tipo de plug-in a criar. (Necessário)

Os plug-ins para um bean que é um elemento-filho do elemento objectGrid são incluídos na lista a seguir:

- Plug-in TransactionCallback
- Plug-in ObjectGridEventListener
- Plug-in SubjectSource
- Plug-in MapAuthorization
- Plug-in SubjectValidation

Os plug-ins válidos para um bean que é um elemento-filho do elemento backingMapPluginCollection são incluídos na lista a seguir:

- Plug-in do Utilitário de Carga
- Plug-in ObjectTransformer
- Plug-in OptimisticCallback
- Plug-in Evictor
- Plug-in MapEventListener
- Plug-in MapIndex

className

Especifica o nome da classe ou o Spring bean para instanciar para criar o plug-in. A classe deve implementar a interface do tipo de plug-in. Por exemplo, se você especificar `ObjectGridEventListener` como o valor do

atributo do ID, o valor do atributo className deve fazer referência a uma classe que implemente a interface ObjectGridEventListener. (Necessário)

```
<bean
(1)   id="TransactionCallback" | "ObjectGridEventListener" |"SubjectSource" |
    "MapAuthorization" | "SubjectValidation" | "Loader" | "ObjectTransformer" |
    "OptimisticCallback" | "Evictor" | "MapEventListener" | "MapIndexPlugin"
(2)   className="class name" | "(spring)bean name"
/>
```

No exemplo a seguir, o arquivo `companyGridBean.xml` é utilizado para demonstrar como configurar plug-ins utilizando o elemento bean. Um plug-in do ObjectGridEventListener é incluído no CompanyGrid ObjectGrid. O atributo className para este bean é a classe com.ibm.websphere.objectgrid.plugins.builtins.TranPropListener. Esta classe implementa a interface com.ibm.websphere.objectgrid.plugins.ObjectGridEventListener conforme necessário.

Um plug-in BackingMap também está definido no arquivo `companyGridBean.xml`. Um plug-in evictor está incluído na instância Customer BackingMap. Como o ID do bean é Evictor, o atributo className deve especificar uma classe que implemente a interface com.ibm.websphere.objectgrid.plugins.Evictor. A classe com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor implementa esta interface. O backingMap faz referência aos seus plug-ins utilizando o atributo pluginCollectionRef.

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

 <objectGrids>
  <objectGrid name="CompanyGrid">
   bean id="ObjectGridEventListener"
    className="com.ibm.websphere.objectgrid.plugins.builtins.TranPropListener"/>
   <backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
  <backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor/>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

A amostra de código a seguir demonstra a abordagem programática para obter a mesma configuração que o arquivo `companyGridBean.xml` no exemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
TranPropListener tranPropListener = new TranPropListener();
companyGrid.addEventListener(tranPropListener);

BackingMap customerMap = companyGrid.defineMap("Customer");
Evictor lruEvictor = new com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor();
customerMap.setEvictor(lruEvictor);
```

Elemento property

Utilize o elemento property para incluir propriedades nos plug-ins. O nome da propriedade deve corresponder a um método configurado na classe referenciada pelo bean de conteúdo.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

Atributos

name
Especifica o nome da propriedade. O valor atribuído a esse atributo deve corresponder a um método set na classe que é fornecida como atributo className no bean que contém o atributo. Por exemplo, se você configurar o atributo className do bean como `com.ibm.MyPlugin` e o nome da propriedade que é fornecido for `size`, a classe `com.ibm.MyPlugin` deve ter um método setSize. (Necessário)

type
Especifica o tipo da propriedade. O tipo é passado para o método configurado que é identificado pelo atributo name. Os valores válidos são os primitivos Java, os correspondentes java.lang e o java.lang.String. Os atributos name e type devem corresponder a uma assinatura de método no atributo className do bean. Por exemplo, se você configurar o nome como `size` e o tipo como `int`, um método setSize(int) deve existir na classe que é especificada como o atributo className para o bean. (Necessário)

value
Especifica o valor da propriedade. Este valor é convertido no tipo especificado pelo atributo type e, em seguida, é utilizado como um parâmetro na chamada para o método set identificado pelos atributos name e type. O valor deste atributo não é validado de nenhuma maneira. (Necessário)

description
Descreve a propriedade. (Opcional)

```
<bean
(1)   name="name"
(2)   type="java.lang.String" | "boolean" | "java.lang.Boolean" | "int" |
    "java.lang.Integer" | "double" | "java.lang.Double" | "byte" |
    "java.lang.Byte" | "short" | "java.lang.Short" | "long" |
    "java.lang.Long" | "float" | "java.lang.Float" | "char" |
    "java.lang.Character"
(3)   value="value"
(4)   description="description"
/>
```

No exemplo a seguir, o arquivo `companyGridProperty.xml` é utilizado para demonstrar como incluir um elemento property em um bean. Neste exemplo, uma propriedade com o nome maxSize e o tipo int são incluídos em um evictor. O Evictor com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor tem uma assinatura de método que corresponde ao método setMaxSize(int). Um valor de número inteiro de 499 é passado para o método setMaxSize(int) na classe com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor.

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

 <objectGrids>
  <objectGrid name="CompanyGrid">
   <backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
  <backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor>
     <property name="maxSize" type="int" value="449"
       description="The maximum size of the LRU Evictor"/>
   </bean>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

A amostra de código a seguir demonstra a abordagem programática para obter a mesma configuração do arquivo companyGridProperty.xml no exemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);

BackingMap customerMap = companyGrid.defineMap("Customer");

LRUEvictor lruEvictor = new com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor();
// if the XML file is used instead,
// the property that was added would cause the following call to occur
lruEvictor.setMaxSize(449);
customerMap.setEvictor(lruEvictor);
```

Elemento backingMapPluginsCollections

O elemento backingMapPluginsCollections é um contêiner para todos os elementos backingMapPluginCollection. No arquivo companyGridProperty.xml na seção anterior, o elemento backingMapPluginCollections contém um elemento backingMapPluginCollection com o ID customerPlugins.

- Número de ocorrências: Zero para uma
- Elemento-filho: Elemento backingMapPluginCollection

Elemento backingMapPluginCollection

O elemento backingMapPluginCollection define os plug-ins do BackingMap e é identificado pelo atributo **id**. Especifique o atributo pluginCollectionRef para referenciar os plug-ins. Ao configurar vários plug-ins do BackingMaps de maneira semelhante, cada BackingMap pode referenciar o mesmo elemento backingMapPluginCollection.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Elemento bean

Atributos

id Identifica a backingMapPluginCollection e é referenciado pelo atributo pluginCollectionRef do elemento backingMap. Cada ID deve ser exclusivo. Se o valor de um atributo pluginCollectionRef não corresponder ao ID de um elemento backingMapPluginCollection, a validação XML falha. Qualquer número de elementos backingMap pode fazer referência a um único elemento backingMapPluginCollection. (Necessário)

```
<backingMapPluginCollection
(1)   id="id"
/>
```

No exemplo a seguir, o arquivo companyGridCollection.xml é utilizado para demonstrar como utilizar o elemento backingMapPluginCollection. Neste arquivo, o BackingMap do Cliente utilizar o customerPlugins backingMapPluginCollection para configurar o BackingMap do Cliente com um LRUEvictor. O Item e OrderLine BackingMaps referenciam o collection2 backingMapPluginCollection. Cada um destes BackingMaps possuem um conjunto de LFUEvictor.

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

 <objectGrids>
  <objectGrid name="CompanyGrid">
   <backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
   <backingMap name="Item" pluginCollectionRef="collection2"/>
   <backingMap name="OrderLine"
    pluginCollectionRef="collection2"/>
```

```
   <backingMap name="Order"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
  <backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor/>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="collection2">
   <bean id="Evictor"
     className="com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor/>
   <bean id="OptimisticCallback"
     className="com.ibm.websphere.samples.objectgrid.EmployeeOptimisticCallBackImpl"/>
  </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

A amostra de código a seguir demonstra a abordagem programática para obter a mesma configuração que o arquivo `companyGridCollection.xml` no exemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();

ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
BackingMap customerMap = companyGrid.defineMap("Customer");
LRUEvictor customerEvictor = new LRUEvictor();
customerMap.setEvictor(customerEvictor);

BackingMap itemMap = companyGrid.defineMap("Item");
LFUEvictor itemEvictor = new LFUEvictor();
itemMap.setEvictor(itemEvictor);

BackingMap orderLineMap = companyGrid.defineMap("OrderLine");
LFUEvictor orderLineEvictor = new LFUEvictor();
orderLineMap.setEvictor(orderLineEvictor);

BackingMap orderMap = companyGrid.defineMap("Order");
```

Elemento querySchema

O elemento querySchema define relacionamentos entre BackingMaps e identifica o tipo de objeto em cada mapa. Estas informações são utilizadas pelo ObjectQuery para converter cadeias de linguagem de consulta em chamadas de acesso de mapa.

- Número de ocorrências: Zero para uma
- Elemento-filho: Elemento mapSchemas, elemento relationships

Elemento mapSchemas

Cada elemento querySchema tem um elemento mapSchemas que contém um ou mais elementos mapSchema.

- Número de ocorrências: Uma
- Elemento-filho: Elemento mapSchema

Elemento mapSchema

Um elemento mapSchema define o tipo de objeto que é armazenado em um BackingMap e instruções sobre como acessar os dados.

- Número de ocorrências: Uma ou mais
- Elemento-filho: Nenhum

Atributos

mapName
Especifica o nome do BackingMap a incluir no esquema. (Necessário)

valueClass
Especifica o tipo de objeto que é armazenado na parte do valor do BackingMap. (Necessário)

primaryKeyField
Especifica o nome do atributo-chave principal no atributo valueClass. A chave primária também deve ser armazenada na parte da chave do BackingMap. (Opcional)

accessType
Identifica como o mecanismo de consulta examina e acessa os dados persistentes nas instâncias do objeto valueClass. Se você configurar o valor como `FIELD`, os campos de classe são examinados e incluídos no esquema. Se o valor for `PROPERTY`, os atributos que estão associados com os métodos get e is são utilizados. O valor padrão é `PROPERTY`. (Opcional)

```
<mapSchema
(1)   mapName="backingMapName"
(2)   valueClass="com.mycompany.OrderBean"
(3)   primaryKeyField="orderId"
(4)   accessType="PROPERTY" | "FIELD"
/>
```

No exemplo a seguir, o arquivo `companyGridQuerySchemaAttr.xml` é utilizado para demonstrar uma configuração do mapSchema de amostra.

```
<?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>
```

A amostra de código a seguir demonstra a abordagem programática para obter a mesma configuração do arquivo `companyGridQuerySchemaAttr.xml` no exemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Definir o esquema
QueryConfig queryCfg = new QueryConfig();
queryCfg.addQueryMapping(new QueryMapping(
    "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryMapping(new QueryMapping(
    "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
companyGrid.setQueryConfig(queryCfg);
```

Elemento relationships

Cada elemento querySchema tem zero ou um elemento relationships que contém um ou mais elementos relationship.

- Número de ocorrências: Zero ou uma
- Elemento-filho: Elemento relationship

Elemento relationship

Um elemento relationship define o relacionamento entre dois BackingMaps e os atributos no atributo valueClass que liga o relacionamento.

- Número de ocorrências: Uma ou mais
- Elemento-filho: Nenhum

Atributos

source
Especifica o nome da valueClass do lado da origem de um relacionamento. (Necessário)

target
Especifica o nome da valueClass do lado do destino de um relacionamento. (Necessário)

relationField
Especifica o nome do atributo na valueClass de origem que faz referência ao destino. (Necessário)

invRelationField
Especifica o nome do atributo na valueClass de destino que faz referência à origem. Se este atributo não for especificado, o relacionamento é unidirecional. (Opcional)

```
<mapSchema
(1)   source="com.mycompany.OrderBean"
(2)   target="com.mycompany.CustomerBean"
(3)   relationField="customer"
(4)   invRelationField="orders"
/>
```

No exemplo a seguir, o arquivo `companyGridQuerySchemaWithRelationshipAttr.xml` é utilizado para demonstrar uma configuração de mapSchema de amostra que inclui um relacionamento bidirecional.

```
<?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>
```

A amostra de código a seguir demonstra a abordagem programática para obter a mesma configuração que o arquivo `companyGridQuerySchemaWithRelationshipAttr.xml` no exemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Definir o esquema
QueryConfig queryCfg = new QueryConfig();
queryCfg.addQueryMapping(new QueryMapping(
    "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryMapping(new QueryMapping(
    "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryRelationship(new QueryRelationship(
     OrderBean.class.getName(), CustomerBean.class.getName(), "customer", "orders"));
companyGrid.setQueryConfig(queryCfg);
```

Elemento streamQuerySet

O elemento streamQuerySet é o elemento de nível superior para definir um conjunto de consultas de fluxo.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Elemento stream, elemento view

Elemento stream

O elemento stream representa um fluxo para o mecanismo de consulta do fluxo. Cada atributo do elemento stream corresponde a um método da interface StreamMetadata.

- Número de ocorrências: Uma para muitas
- Elemento-filho: Elemento basic

Atributos

name
Especifica o nome do fluxo. A validação falha se este atributo não for especificado. (Necessário)

valueClass
Especifica o tipo de classe do valor que é armazenado no ObjectMap do fluxo. O tipo de classe é utilizado para converter o objeto para os eventos de fluxo e para gerar uma instrução SQL se a instrução não for fornecida. (Necessário)

sql
Especifica a instrução SQL do fluxo. Se essa propriedade não for fornecida, um SQL de fluxo será gerado, refletindo os atributos ou métodos do acessador no atributo valueClass ou utilizando os atributos tuple do metadados da entidade. (Opcional)

access
Especifica o tipo para acessar os atributos da classe de valor. Se você configurar o valor para `FIELD`, os atributos serão recuperados diretamente a

partir dos campos usando o reflexo Java. Caso contrário, os métodos do acessador serão utilizados para ler os atributos. O valor padrão é `PROPERTY`. (Opcional)

```
<stream
(1)   name="streamName"
(2)   valueClass="streamMapClassType"
(3)   sql="streamSQL create stream stockQuote
   keyed by t ( transactionvolume INTEGER,
                        price DECIMAL (9,2), issue VARCHAR(100) );"
(4)   access="PROPERTY" | "FIELD"
/>
```

Elemento view

O elemento view representa uma visualização de consulta de fluxo. Cada elemento stream corresponde a um método na interface ViewMetadata.

- Número de ocorrências: Uma para muitas
- Elemento-filho: Elemento basic, elemento id

Atributos

name
Especifica o nome da visualização. A validação falha se este atributo não for especificado. (Necessário)

sql
Especifica o SQL do fluxo, que define a transformação da visualização. A validação falha se este atributo não for especificado. (Necessário)

valueClass
Especifica o tipo de classe do valor que é armazenado nesta visualização do ObjectMap. O tipo de classe é utilizado para converter eventos de visualização no formato de tupla correto que é compatível com este tipo de classe. Se o tipo de classe não for fornecido, um formato padrão após as definições da coluna em SPTSQL (Stream Processing Technology Structured Query Language) será utilizado. Se um metadado de entidade é definido para este mapa de visualização, este atributo não deve ser utilizado. O metadado da entidade é utilizado em seu lugar. (Opcional)

access
Especifica o tipo para acessar os atributos da classe de valor. Se você configurar o tipo de acesso para `FIELD`, os valores de coluna serão configurados diretamente para os campos usando o reflexo Java. Caso contrário, os métodos do acessador são utilizados para configurar os atributos. O valor padrão é `PROPERTY`. (Opcional)

```
<view
(1)   name="viewName"
(2)   valueClass="viewMapValueClass"
(3)   sql="viewSQL CREATE VIEW last5MinuteAvgPrice AS
      SELECT issue, avg(price) as totalVolume
        FROM (SELECT * FROM stockQuote FETCH LATEST 5 MINUTES) group by issue;"/>
(4)   access="PROPERTY" | "FIELD"
/>
```

Elemento basic

O elemento basic é utilizado para definir um mapeamento a partir do nome do atributo na classe de valor ou metadados da entidade para a coluna que é definida no SPTSQL.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

```
<basic
(1)   name="attributeName"
(2)   column="columnName"
/>
```

Elemento ID

O elemento id é utilizado para um mapeamento de atributo-chave.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

```
<id
(1)   name="idName"
(2)   column="columnName"
/>
```

No exemplo a seguir, o arquivo `StreamQueryApp2.xml` é utilizado para demonstrar como configurar os atributos de um streamQuerySet. O conjunto de consultas de fluxo _stockQuoteSQS_ possui um fluxo e uma visualização. O fluxo e a visualização definem seu nome, valueClasss, sql e tipo de acesso respectivamente. O fluxo também define um elemento básico, que especifica que o atributo volume na classe StockQuote é mapeado para o volume de transação da coluna SQL que é definida na instrução SQL.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
 <objectGrids>
  <objectGrid name="og1">
   <backingMap name="stockQuote" readOnly="false" copyKey="true" streamRef="stockQuote"/>
   <backingMap name="last5MinuteAvgPrice" readOnly="false" copyKey="false" viewRef="last5MinuteAvgPrice"/>

   <streamQuerySet name="stockQuoteSQS">
    <stream
     name="stockQuote"
     valueClass="com.ibm.ws.objectgrid.streamquery.sample.guide.StockQuote"
     sql="create stream stockQuote
     keyed by t ( transactionvolume INTEGER,
                                    price DECIMAL (9,2), issue VARCHAR(100) );"
     access="FIELD">
     <basic name="volume" column="transactionvolume"/>
    </stream>

    <view
     name="last5MinuteAvgPrice"
     valueClass="com.ibm.ws.objectgrid.streamquery.sample.guide.AveragePrice"
     sql="CREATE VIEW last5MinuteAvgPrice AS SELECT issue, avg(price) as avgPrice
       FROM (SELECT * FROM stockQuote FETCH LATEST 5 MINUTES) group by issue;"
     access="FIELD"
    </view>
   </streamQuerySet>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Arquivo `objectGrid.xsd`

Use o esquema XML do descritor do ObjectGrid para configurar o WebSphere eXtreme Scale.

Consulte o “Arquivo XML descritor do ObjectGrid” na página 150 para obter descrições dos elementos e atributos definidos no arquivo `objectGrid.xsd`. Para obter mais informações sobre o arquivo `objectgrid.xsd` Spring, consulte “Arquivo XML descritor do Spring” na página 206.

```
<?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>
```

Arquivo Descritor XML de Metadados de Entidade

O arquivo descritor de metadados da entidade é um arquivo XML que é utilizado para definir um esquema de entidade para o WebSphere eXtreme Scale. Defina todos os metadados de entidade no arquivo XML ou defina os metadados de entidade como anotações no arquivo de classe Java de entidade. O uso principal é para entidades que não podem usar as anotações Java.

Utilize a configuração do XML para criar os metadados de entidade com base no arquivo XML. Quando utilizado em conjunto com a anotação, alguns dos atributos definidos na configuração do XML sobrescrevem as anotações correspondentes. Se for possível substituir um elemento, a substituição estará explicitamente nas seções a seguir. Consulte o “Arquivo `emd.xsd`” na página 182 para obter um exemplo do arquivo descritor XML de metadados da entidade.

Elemento ID

O elemento id implica que o atributo é uma chave. Pelo menos, um elemento id deve ser especificado. Você pode especificar várias chaves de id para serem utilizadas como chave composta.

Atributos

name
Especifica o nome do atributo. O atributo deve existir no arquivo Java.

alias
Especifica o alias do elemento. O valor do alias é substituído se utilizado em conjunto com uma entidade anotada.

Elemento basic

O elemento basic implica que o atributo é um tipo primitivo ou wrappers para tipos primitivos:

- java.lang.String
- java.math.BigInteger
- java.math.BigDecimal
- java.util.Date
- java.util.Calendar
- java.sql.Date
- java.sql.Time
- java.sql.Timestamp
- byte[]
- Byte[]
- char[]
- Character[]
- Enum Java Platform, Standard Edition Versão 5

É necessário especificar qualquer atributo como básico. Os atributos element básicos são automaticamente configurados utilizando reflexo.

Elemento id-class

O elemento id_class especifica uma classe de chave composta, que ajuda a localizar entidades com chaves compostas.

Atributos

class-name
Especifica o nome da classe, que é um id-class, para uso com o elemento id-class.

transient

O elemento transient implica que ele é ignorado e não processado. Ele também pode ser substituído se utilizado em conjunto com entidades anotadas.

Atributos

name
Especifica o nome do atributo, que é ignorado.

versão

O elemento transient implica que ele é ignorado e não processado. Ele também pode ser substituído se utilizado em conjunto com entidades anotadas.

Atributos

name
Especifica o nome do atributo, que é ignorado.

Elemento property

Utilize o elemento property para incluir propriedades nos plug-ins. O nome da propriedade deve corresponder a um método configurado na classe referenciada pelo bean de conteúdo.

- Número de ocorrências: Zero para muitas

- Elemento-filho: Nenhum

Atributos

name
Especifica o nome da propriedade. O valor atribuído a esse atributo deve corresponder a um método set na classe que é fornecida como atributo className no bean que contém o atributo. Por exemplo, se você configurar o atributo className do bean como `com.ibm.MyPlugin` e o nome da propriedade que é fornecido for `size`, a classe `com.ibm.MyPlugin` deve ter um método setSize. (Necessário)

type
Especifica o tipo da propriedade. O tipo é passado para o método configurado que é identificado pelo atributo name. Os valores válidos são os primitivos Java, os correspondentes java.lang e o java.lang.String. Os atributos name e type devem corresponder a uma assinatura de método no atributo className do bean. Por exemplo, se você configurar o nome como `size` e o tipo como `int`, um método setSize(int) deve existir na classe que é especificada como o atributo className para o bean. (Necessário)

value
Especifica o valor da propriedade. Este valor é convertido no tipo especificado pelo atributo type e, em seguida, é utilizado como um parâmetro na chamada para o método set identificado pelos atributos name e type. O valor deste atributo não é validado de nenhuma maneira. (Necessário)

description
Descreve a propriedade. (Opcional)

```
<bean
(1)   name="name"
(2)   type="java.lang.String" | "boolean" | "java.lang.Boolean" | "int" |
       "java.lang.Integer" | "double" | "java.lang.Double" | "byte" |
       "java.lang.Byte" | "short" | "java.lang.Short" | "long" |
       "java.lang.Long" | "float" | "java.lang.Float" | "char" |
       "java.lang.Character"
(3)   value="value"
(4)   description="description"
/>
```

No exemplo a seguir, o arquivo `companyGridProperty.xml` é utilizado para demonstrar como incluir um elemento property em um bean. Neste exemplo, uma propriedade com o nome maxSize e o tipo int são incluídos em um evictor. O Evictor com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor tem uma assinatura de método que corresponde ao método setMaxSize(int). Um valor de número inteiro de 499 é passado para o método setMaxSize(int) na classe com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor.

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

 <objectGrids>
  <objectGrid name="CompanyGrid">
   <backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
  <backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor>
     <property name="maxSize" type="int" value="449"
       description="The maximum size of the LRU Evictor"/>
```

```
   </bean>
  </backingMapPluginCollection>
 </backingMapPluginCollections>
</objectGridConfig>
```

A amostra de código a seguir demonstra a abordagem programática para obter a mesma configuração do arquivo `companyGridProperty.xml` no exemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);

BackingMap customerMap = companyGrid.defineMap("Customer");

LRUEvictor lruEvictor = new com.ibm.websphere.objectgrid.plugins.builtins.LRUEvictor();
// if the XML file is used instead,
// the property that was added would cause the following call to occur
lruEvictor.setMaxSize(449);
customerMap.setEvictor(lruEvictor);
```

Elemento backingMapPluginsCollections

O elemento backingMapPluginsCollections é um contêiner para todos os elementos backingMapPluginCollection. No arquivo `companyGridProperty.xml` na seção anterior, o elemento backingMapPluginCollections contém um elemento backingMapPluginCollection com o ID `customerPlugins`.

- Número de ocorrências: Zero para uma
- Elemento-filho: Elemento backingMapPluginCollection

Elemento backingMapPluginCollection

O elemento backingMapPluginCollection define os plug-ins do BackingMap e é identificado pelo atributo **id**. Especifique o atributo pluginCollectionRef para referenciar os plug-ins. Ao configurar vários plug-ins do BackingMaps de maneira semelhante, cada BackingMap pode referenciar o mesmo elemento backingMapPluginCollection.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Elemento bean

Atributos

id Identifica a backingMapPluginCollection e é referenciado pelo atributo pluginCollectionRef do elemento backingMap. Cada ID deve ser exclusivo. Se o valor de um atributo pluginCollectionRef não corresponder ao ID de um elemento backingMapPluginCollection, a validação XML falha. Qualquer número de elementos backingMap pode fazer referência a um único elemento backingMapPluginCollection. (Necessário)

```
<backingMapPluginCollection
(1)   id="id"
/>
```

No exemplo a seguir, o arquivo `companyGridCollection.xml` é utilizado para demonstrar como utilizar o elemento backingMapPluginCollection. Neste arquivo, o BackingMap do Cliente utilizar o customerPlugins backingMapPluginCollection para configurar o BackingMap do Cliente com um LRUEvictor. O Item e OrderLine BackingMaps referenciam o collection2 backingMapPluginCollection. Cada um destes BackingMaps possuem um conjunto de LFUEvictor.

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

 <objectGrids>
```

```
  <objectGrid name="CompanyGrid">
   <backingMap name="Customer"
    pluginCollectionRef="customerPlugins"/>
   <backingMap name="Item" pluginCollectionRef="collection2"/>
   <backingMap name="OrderLine"
    pluginCollectionRef="collection2"/>
   <backingMap name="Order"/>
  </objectGrid>
 </objectGrids>
 <backingMapPluginCollections>
  <backingMapPluginCollection id="customerPlugins">
   <bean id="Evictor"
     className="com.ibm.websphere.objectGrid.plugins.builtins.LRUEvictor/>
  </backingMapPluginCollection>
  <backingMapPluginCollection id="collection2">
   <bean id="Evictor"
     className="com.ibm.websphere.objectgrid.plugins.builtins.LFUEvictor/>
   <bean id="OptimisticCallback"
     className="com.ibm.websphere.samples.objectgrid.EmployeeOptimisticCallBackImpl"/>
  </backingMapPluginCollection>
  </backingMapPluginCollections>
</objectGridConfig>
```

A amostra de código a seguir demonstra a abordagem programática para obter a mesma configuração que o arquivo `companyGridCollection.xml` no exemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();

ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
BackingMap customerMap = companyGrid.defineMap("Customer");
LRUEvictor customerEvictor = new LRUEvictor();
customerMap.setEvictor(customerEvictor);

BackingMap itemMap = companyGrid.defineMap("Item");
LFUEvictor itemEvictor = new LFUEvictor();
itemMap.setEvictor(itemEvictor);

BackingMap orderLineMap = companyGrid.defineMap("OrderLine");
LFUEvictor orderLineEvictor = new LFUEvictor();
orderLineMap.setEvictor(orderLineEvictor);

BackingMap orderMap = companyGrid.defineMap("Order");
```

Elemento querySchema

O elemento querySchema define relacionamentos entre BackingMaps e identifica o tipo de objeto em cada mapa. Estas informações são utilizadas pelo ObjectQuery para converter cadeias de linguagem de consulta em chamadas de acesso de mapa. Para obter mais informações, consulte os detalhes sobre a definição de um esquema ObjectQuery no *Guia de Programação*.

- Número de ocorrências: Zero para uma
- Elemento-filho: Elemento mapSchemas, elemento relationships

Elemento mapSchemas

Cada elemento querySchema tem um elemento mapSchemas que contém um ou mais elementos mapSchema.

- Número de ocorrências: Uma
- Elemento-filho: Elemento mapSchema

Elemento mapSchema

Um elemento mapSchema define o tipo de objeto que é armazenado em um BackingMap e instruções sobre como acessar os dados.

- Número de ocorrências: Uma ou mais

- Elemento-filho: Nenhum

Atributos

mapName
Especifica o nome do BackingMap a incluir no esquema. (Necessário)

valueClass
Especifica o tipo de objeto que é armazenado na parte do valor do BackingMap. (Necessário)

primaryKeyField
Especifica o nome do atributo-chave principal no atributo valueClass. A chave primária também deve ser armazenada na parte da chave do BackingMap. (Opcional)

accessType
Identifica como o mecanismo de consulta examina e acessa os dados persistentes nas instâncias do objeto valueClass. Se você configurar o valor como `FIELD`, os campos de classe são examinados e incluídos no esquema. Se o valor for `PROPERTY`, os atributos que estão associados com os métodos get e is são utilizados. O valor padrão é `PROPERTY`. (Opcional)

```
<mapSchema
(1)   mapName="backingMapName"
(2)   valueClass="com.mycompany.OrderBean"
(3)   primaryKeyField="orderId"
(4)   accessType="PROPERTY" | "FIELD"
/>
```

No exemplo a seguir, o arquivo `companyGridQuerySchemaAttr.xml` é utilizado para demonstrar uma configuração do mapSchema de amostra.

```
<?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>
```

A amostra de código a seguir demonstra a abordagem programática para obter a mesma configuração do arquivo `companyGridQuerySchemaAttr.xml` no exemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Definir o esquema
QueryConfig queryCfg = new QueryConfig();
queryCfg.addQueryMapping(new QueryMapping(
```

```
    "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryMapping(new QueryMapping(
    "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
companyGrid.setQueryConfig(queryCfg);
```

Elemento relationships

Cada elemento querySchema tem zero ou um elemento relationships que contém um ou mais elementos relationship.

- Número de ocorrências: Zero ou uma
- Elemento-filho: Elemento relationship

Elemento relationship

Um elemento relationship define o relacionamento entre dois BackingMaps e os atributos no atributo valueClass que liga o relacionamento.

- Número de ocorrências: Uma ou mais
- Elemento-filho: Nenhum

Atributos

source
Especifica o nome da valueClass do lado da origem de um relacionamento. (Necessário)

target
Especifica o nome da valueClass do lado do destino de um relacionamento. (Necessário)

relationField
Especifica o nome do atributo na valueClass de origem que faz referência ao destino. (Necessário)

invRelationField
Especifica o nome do atributo na valueClass de destino que faz referência à origem. Se este atributo não for especificado, o relacionamento é unidirecional. (Opcional)

```
<mapSchema
(1)   source="com.mycompany.OrderBean"
(2)   target="com.mycompany.CustomerBean"
(3)   relationField="customer"
(4)   invRelationField="orders"
/>
```

No exemplo a seguir, o arquivo `companyGridQuerySchemaWithRelationshipAttr.xml` é utilizado para demonstrar uma configuração de mapSchema de amostra que inclui um relacionamento bidirecional.

```
<?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>
```

A amostra de código a seguir demonstra a abordagem programática para obter a mesma configuração que o arquivo `companyGridQuerySchemaWithRelationshipAttr.xml` no exemplo anterior.

```
ObjectGridManager objectGridManager = ObjectGridManagerFactory.getObjectGridManager();
ObjectGrid companyGrid = objectGridManager.createObjectGrid("CompanyGrid", false);
companyGrid.defineMap("Order");
companyGrid.defineMap("Customer");

// Definir o esquema
QueryConfig queryCfg = new QueryConfig();
queryCfg.addQueryMapping(new QueryMapping(
    "Order", OrderBean.class.getName(), "orderNumber", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryMapping(new QueryMapping(
    "Customer", CustomerBean.class.getName(), "id", QueryMapping.FIELD_ACCESS));
queryCfg.addQueryRelationship(new QueryRelationship(
     OrderBean.class.getName(), CustomerBean.class.getName(), "customer", "orders"));
companyGrid.setQueryConfig(queryCfg);
```

Elemento timeBasedDBUpdate

Um elemento timeBasedDBUpdate define uma configuração para um atualizador de banco de dados baseado em tempo. Um elemento timeBasedDBUpdate contém informações sobre com que frequência os registros inseridos e atualizados recentemente são recuperados a partir do banco de dados usando o Java Persistence API (JPA) e sobre como atualizar os dados nos mapas ObjectGrid correspondentes.

- Número de ocorrências: Zero ou uma
- Elemento-filho: Nenhum

Atributos

entityClass
Especifica o nome da classe de entidade utilizado para interagir com o provedor JPA. O nome da classe de entidade é utilizado para recuperar entidades JPA utilizando consultas de entidade. (Necessário)

persistenceUnitName
Especifica o nome da unidade de persistência JPA para criação de um factory do gerenciador de entidades JPA. O valor padrão é o nome da primeira unidade de persistência definida no arquivo `persistence.xml`. (Opcional)

mode
Especifica o modo de atualização de banco de dados baseado em tempo. Por padrão, o modo de atualização de banco de dados baseado em tempo é configurado como INVALIDATE_ONLY. Um tipo INVALIDATE_ONLY indica a invalidação das entradas no mapa do ObjectGrid se os registros correspondentes no banco de dados foram alterados. Um tipo UPDATE_ONLY indica a atualização das entradas existentes no mapa do ObjectGrid com os valores mais recentes do banco de dados. Entretanto, todos os registros recentemente inseridos no banco de dados são ignorados. Um tipo INSERT_UPDATE indica a atualização das entradas existentes no mapa do

ObjectGrid com os valores mais recentes do banco de dados. Além disso, todos os registros recentemente inseridos no banco de dados são inseridos no mapa do ObjectGrid. (Opcional)

timestampField
Especifica o nome do campo do registro de data e hora. Um valor de campo de registro de data e hora é utilizado para identificar a hora ou sequência quando um registro de backend de banco de dados foi utilizado pela última vez. (Opcional)

jpaPropertyFactory
Identifica o nome da classe de implementação do JPAPropertyFactory ou Spring bean. A interface com.ibm.websphere.objectgrid.jpa.JPAPropertyFactory é utilizada para conectar a mapa da propriedade de persistência para substituir as propriedades JPA padrão. Utilize os beans de estrutura spring se for necessário configurar atributos adicionais na instância do JPAPropertyFactory. Consulte "Integração com a Estrutura Spring" na página 200 para obter mais informações. (Opcional)

```
<timeBasedDBUpdate
(1)   persistenceUnitName="SamplePU"
(2)   mode="INVALIDATE_ONLY" | "UPDATE_ONLY" | "INSERT_UPDATE"
(3)   timestampField="TIMESTAMP"
(4)   entityClass="entity class"
(5)   jpaPropertyFactory="JPA property factory class" | "{spring}bean name"
/>
```

Elemento streamQuerySet

O elemento streamQuerySet é o elemento de nível superior para definir um conjunto de consultas de fluxo.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Elemento stream, elemento view

Elemento stream

O elemento stream representa um fluxo para o mecanismo de consulta do fluxo. Cada atributo do elemento stream corresponde a um método da interface StreamMetadata.

- Número de ocorrências: Uma para muitas
- Elemento-filho: Elemento basic

Atributos

name
Especifica o nome do fluxo. A validação falha se este atributo não for especificado. (Necessário)

valueClass
Especifica o tipo de classe do valor que é armazenado no ObjectMap do fluxo. O tipo de classe é utilizado para converter o objeto para os eventos de fluxo e para gerar uma instrução SQL se a instrução não for fornecida. (Necessário)

sql
Especifica a instrução SQL do fluxo. Se essa propriedade não for fornecida, um SQL de fluxo será gerado, refletindo os atributos ou métodos do acessador no atributo valueClass ou utilizando os atributos tuple do metadados da entidade. (Opcional)

access
Especifica o tipo para acessar os atributos da classe de valor. Se você configurar o valor para `FIELD`, os atributos serão recuperados diretamente a partir dos campos usando o reflexo Java. Caso contrário, os métodos do acessador serão utilizados para ler os atributos. O valor padrão é `PROPERTY`. (Opcional)

```
<stream
(1)   name="streamName"
(2)   valueClass="streamMapClassType"
(3)   sql="streamSQL create stream stockQuote
       keyed by t ( transactionvolume INTEGER,
                    price DECIMAL (9,2), issue VARCHAR(100) );"
(4)   access="PROPERTY" | "FIELD"
/>
```

Elemento view

O elemento view representa uma visualização de consulta de fluxo. Cada elemento stream corresponde a um método na interface ViewMetadata.

- Número de ocorrências: Uma para muitas
- Elemento-filho: Elemento basic, elemento id

Atributos

name
Especifica o nome da visualização. A validação falha se este atributo não for especificado. (Necessário)

sql
Especifica o SQL do fluxo, que define a transformação da visualização. A validação falha se este atributo não for especificado. (Necessário)

valueClass
Especifica o tipo de classe do valor que é armazenado nesta visualização do ObjectMap. O tipo de classe é utilizado para converter eventos de visualização no formato de tupla correto que é compatível com este tipo de classe. Se o tipo de classe não for fornecido, um formato padrão após as definições da coluna em SPTSQL (Stream Processing Technology Structured Query Language) será utilizado. Se um metadado de entidade é definido para este mapa de visualização, este atributo não deve ser utilizado. O metadado da entidade é utilizado em seu lugar. (Opcional)

access
Especifica o tipo para acessar os atributos da classe de valor. Se você configurar o tipo de acesso para `FIELD`, os valores de coluna serão configurados diretamente para os campos usando o reflexo Java. Caso contrário, os métodos do acessador são utilizados para configurar os atributos. O valor padrão é `PROPERTY`. (Opcional)

```
<view
(1)   name="viewName"
(2)   valueClass="viewMapValueClass"
(3)   sql="viewSQL CREATE VIEW last5MinuteAvgPrice AS
      SELECT issue, avg(price) as totalVolume
        FROM (SELECT * FROM stockQuote FETCH LATEST 5 MINUTES) group by issue;"/>
(4)   access="PROPERTY" | "FIELD"
/>
```

Elemento basic

O elemento basic é utilizado para definir um mapeamento a partir do nome do atributo na classe de valor ou metadados da entidade para a coluna que é definida no SPTSQL.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

Atributos

```
<basic
(1)   name="attributeName"
(2)   column="columnName"
/>
```

Elemento ID

O elemento id é utilizado para um mapeamento de atributo-chave.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

Atributos

```
<id
(1)   name="idName"
(2)   column="columnName"
/>
```

No exemplo a seguir, o arquivo `StreamQueryApp2.xml` é utilizado para demonstrar como configurar os atributos de um streamQuerySet. O conjunto de consultas de fluxo _stockQuoteSQS_ possui um fluxo e uma visualização. O fluxo e a visualização definem o nome, valueClass, sql e o tipo de acesso respectivamente. O fluxo também define um elemento básico, que especifica que o atributo volume na classe StockQuote é mapeado para o transactionvolume da coluna SQL que é definida na instrução SQL.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
 <objectGrids>
  <objectGrid name="og1">
   <backingMap name="stockQuote" readOnly="false" copyKey="true" streamRef="stockQuote"/>
   <backingMap name="last5MinuteAvgPrice" readOnly="false" copyKey="false" viewRef="last5MinuteAvgPrice"/>

   <streamQuerySet name="stockQuoteSQS">
    <stream
     name="stockQuote"
     valueClass="com.ibm.ws.objectgrid.streamquery.sample.guide.StockQuote"
     sql="create stream stockQuote
     keyed by t ( transactionvolume INTEGER,
                                    price DECIMAL (9,2), issue VARCHAR(100) );"
     access="FIELD">
     <basic name="volume" column="transactionvolume"/>
    </stream>

    <view
     name="last5MinuteAvgPrice"
     valueClass="com.ibm.ws.objectgrid.streamquery.sample.guide.AveragePrice"
     sql="CREATE VIEW last5MinuteAvgPrice AS SELECT issue, avg(price) as avgPrice
       FROM (SELECT * FROM stockQuote FETCH LATEST 5 MINUTES) group by issue;"
     access="FIELD"
    </view>
   </streamQuerySet>
  </objectGrid>
 </objectGrids>
</objectGridConfig>
```

Arquivo emd.xsd

Use a definição de esquema XML de metadados de entidade para criar um arquivo descritor XML e definir um esquema de entidade para o WebSphere eXtreme Scale.

Consulte “Arquivo Descritor XML de Metadados de Entidade” na página 171 para obter descrições de cada elemento e atributo do arquivo emd.xsd.

Arquivo 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>

  1 <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>
```

Arquivo XML Descritor de Segurança

Use um arquivo descritor de segurança do ObjectGrid para configurar uma topologia de implementação do eXtreme Scale com segurança ativada. Amostras de arquivos XML são fornecidas neste tópico para ilustrar várias configurações.

Cada elemento e atributo do arquivo XML do cluster está descrito na lista a seguir. Utilize os exemplos para aprender como utilizar esses elementos e atributos para configurar o ambiente.

Elemento securityConfig

O elemento securityConfig é o elemento de nível superior do arquivo XML de segurança do ObjectGrid. Este elemento configura o espaço de nomes do arquivo e o local do esquema. O esquema está definido no arquivo `objectGridSecurity.xsd`.

- Número de ocorrências: Uma
- Elementos filho : segurança

Elemento security

Utilize o elemento security para definir uma segurança do ObjectGrid.

- Número de ocorrências: Uma
- Elementos-filho: authenticator, adminAuthorization e systemCredentialGenerator

Atributos

securityEnabled
Ativa a segurança para a grade quando configurado para true. O valor padrão é false Se o valor for configurado como false, a segurança no escopo da grade será desativada. Para obter informações adicionais, consulte “Segurança de Grade” na página 286. (Opcional)

singleSignOnEnabled
Permite que o cliente se conecte a qualquer servidor depois que ele for autenticado com um dos servidores, se o valor for configurado para true. Caso contrário, um cliente deverá se autenticar com cada servidor antes de poder se conectar.O valor padrão é false. (Opcional)

loginSessionExpirationTime
Especifica a quantidade de tempo, em segundos, antes de a sessão de login expirar. Se a sessão de login expirar, o cliente precisa autenticar-se novamente. (Opcional)

adminAuthorizationEnabled
Ativa a autorização administrativa. Se o valor for configurado para true, todas as tarefas administrativas precisarão de autorização. O mecanismo de autorização utilizado é especificado pelo valor do atributo adminAuthorizationMechanism. O valor padrão é false. (Opcional)

adminAuthorizationMechanism
Indica qual mecanismo de autorização deve ser usado. O WebSphere eXtreme Scale suporta dois mecanismos de autorização, JAAS (Java Authentication and Authorization Service) e autorização personalizada. O mecanismo de autorização JAAS utiliza a abordagem baseada em política JAAS padrão. Para especificar JAAS como o mecanismo de autorização, configure o valor como AUTHORIZATION_MECHANISM_JAAS. O mecanismo de autorização customizado utiliza uma implementação de AdminAuthorization conectada pelo usuário. Para especificar um mecanismo de autorização customizado, configure o valor como AUTHORIZATION_MECHANISM_CUSTOM. Para obter informações adicionais sobre como estes dois mecanismos são utilizados, consulte “Autorização do Aplicativo Cliente” na página 290. (Opcional)

O arquivo `security.xml` a seguir é uma configurações de amostra para ativar a segurança da grade do eXtreme Scale.

security.xml

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

    <security securityEnabled="true" singleSignOnEnabled="true"
        loginSessionExpirationTime="20"
        adminAuthorizationEnabled="true"
        adminAuthorizationMechanism="AUTHORIZATION_MECHANISM_JAAS" >

        <authenticator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator">
        </authenticator>

        <systemCredentialGenerator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator">
            <property name="properties" type="java.lang.String" value="runAs" description="Using runAs subject" />
        </systemCredentialGenerator>

    </security>
</securityConfig>
```

Elemento authenticator

Autentica clientes para servidores eXtreme Scale na grade. A classe especificada pelo atributo className deve implementar a interface com.ibm.websphere.objectgrid.security.plugins.Authenticator. O autenticador pode utilizar propriedades para chamar métodos na classe que é especificada pelo

atributo className. Consulte o elemento de propriedade para obter mais informações sobre como utilizar as propriedades.

No exemplo do arquivo `security.xml` anterior, a classe com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator é especificada como o autenticador. Esta classe implementa a interface com.ibm.websphere.objectgrid.security.plugins.Authenticator.

- Número de ocorrências: zero ou uma
- Elemento filho: propriedade

Atributos

className
Especifica a classe que implementa a interface com.ibm.websphere.objectgrid.security.plugins.Authenticator. Use esta classe para autenticar clientes para os servidores na grade do eXtreme Scale. (Obrigatório)

Elemento adminAuthorization

Use o elemento adminAuthorization para configurar o acesso administrativo para a grade.

- Número de ocorrências: zero ou uma
- Elemento filho: propriedade

Atributos

className
Especifica a classe que implementa a interface com.ibm.websphere.objectgrid.security.plugins.AdminAuthorization. (Obrigatório)

Elemento systemCredentialGenerator

Utilize um elemento systemCredentialGenerator para configurar um gerador de credenciais do sistema. Este elemento é aplicável apenas a um ambiente dinâmico. No modelo de configuração dinâmica, o servidor de contêineres dinâmicos se conecta ao servidor de catálogos como um cliente do eXtreme Scale e o servidor de catálogos pode se conectar ao servidor de contêineres do eXtreme Scale como um cliente também. O gerador de credenciais do sistema é utilizado para representar um depósito de informações para a credencial do sistema.

- Número de ocorrências: zero ou uma
- Elemento filho: propriedade

Atributos

className
Especifica a classe que implementa a interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. (Obrigatório)

Consulte o arquivo `security.xml` anterior para saber como utilizar um systemCredentialGenerator. Neste exemplo, o gerador de credenciais do sistema é um com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenCredentialGenerator, que recupera o objeto RunAs Subject a partir do encadeamento.

Elemento property

Chama o método set nas classes authenticator e adminAuthorization. O nome da propriedade corresponde a um método configurado no atributo className do elemento authenticator ou adminAuthorization.

- Número de ocorrências: zero ou mais
- Elemento filho: propriedade

Atributos

name
Especifica o nome da propriedade. O valor atribuído a esse atributo deve corresponder a um método set na classe que é fornecida como atributo className no bean que contém o atributo. Por exemplo, se o atributo className do bean for configurado como com.ibm.MyPlugin e o nome da propriedade que é fornecido for size, então, a classe com.ibm.MyPlugin deve ter um método setSize. (Obrigatório)

type
Especifica o tipo da propriedade. O tipo do parâmetro é transmitido ao método set identificado pelo atributo name. Os valores válidos são as primitivas Java, suas partes java.lang e java.lang.String. Os atributos name e type devem corresponder a uma assinatura de método no atributo className do bean. Por exemplo, se o nome for size e o tipo for int, então, deve existir um método setSize(int) na classe que é especificada como o atributo className para o bean. (Obrigatório)

value
Especifica o valor da propriedade. Este valor é convertido no tipo especificado pelo atributo type e, em seguida, é utilizado como um parâmetro na chamada para o método set identificado pelos atributos name e type. O valor deste atributo não é validado de nenhuma maneira. O implementador do plug-in deve verificar se o valor transmitido é válido. (Obrigatório)

description
Fornece uma descrição da propriedade (Opcional)

Consulte "Arquivo `objectGridSecurity.xsd`" para obter mais informações.

Arquivo `objectGridSecurity.xsd`

Use o seguinte esquema XML de segurança do ObjectGrid para ativar a segurança para uma implementação do eXtreme Scale.

Consulte o "Arquivo XML Descritor de Segurança" na página 184 para obter descrições dos elementos e atributos definidos no arquivo `objectGridSecurity.xsd`.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:cc="http://ibm.com/ws/objectgrid/config/security"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://ibm.com/ws/objectgrid/config/security"
 elementFormDefault="qualified">

 <xsd:element name="securityConfig">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="security" type="cc:security" />
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>

 <xsd:complexType name="security">
  <xsd:sequence>
   <xsd:element name="authenticator" type="cc:bean" minOccurs="0"
    maxOccurs="1" />
```

```
  <xsd:element name="adminAuthorization" type="cc:bean" minOccurs="0"
   maxOccurs="1" />
  <xsd:element name="systemCredentialGenerator" type="cc:bean" minOccurs="0"
   maxOccurs="1" />
 </xsd:sequence>
 <xsd:attribute name="securityEnabled" type="xsd:boolean" use="optional" />
 <xsd:attribute name="singleSignOnEnabled" type="xsd:boolean" use="optional"/>
 <xsd:attribute name="loginSessionExpirationTime" type="xsd:int" use="optional"/>
 <xsd:attribute name="adminAuthorizationMechanism"
              type="cc:adminAuthorizationMechanism" use="optional"/>
 <xsd:attribute name="adminAuthorizationEnabled" type="xsd:boolean" use="optional" />
</xsd:complexType>

<xsd:complexType name="bean">
 <xsd:sequence>
  <xsd:element name="property" type="cc:property" maxOccurs="unbounded" minOccurs="0" />
 </xsd:sequence>
 <xsd:attribute name="className" type="xsd:string" use="required" />
</xsd:complexType>

<xsd:complexType name="property">
 <xsd:attribute name="name" type="xsd:string" use="required" />
 <xsd:attribute name="value" type="xsd:string" use="required" />
 <xsd:attribute name="type" type="cc:propertyType" use="required" />
 <xsd:attribute name="description" type="xsd:string" use="optional" />
</xsd:complexType>

<xsd:simpleType name="propertyType">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="java.lang.Boolean"/>
  <xsd:enumeration value="boolean"/>
  <xsd:enumeration value="java.lang.String"/>
  <xsd:enumeration value="java.lang.Integer"/>
  <xsd:enumeration value="int"/>
  <xsd:enumeration value="java.lang.Double"/>
  <xsd:enumeration value="double"/>
  <xsd:enumeration value="java.lang.Byte"/>
  <xsd:enumeration value="byte"/>
  <xsd:enumeration value="java.lang.Short"/>
  <xsd:enumeration value="short"/>
  <xsd:enumeration value="java.lang.Long"/>
  <xsd:enumeration value="long"/>
  <xsd:enumeration value="java.lang.Float"/>
  <xsd:enumeration value="float"/>
  <xsd:enumeration value="java.lang.Character"/>
  <xsd:enumeration value="char"/>
 </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="adminAuthorizationMechanism">
 <xsd:restriction base="xsd:string">
  <xsd:enumeration value="AUTHORIZATION_MECHANISM_JAAS" />
  <xsd:enumeration value="AUTHORIZATION_MECHANISM_CUSTOM" />
 </xsd:restriction>
</xsd:simpleType>

</xsd:schema>
```

Referência de Arquivo de Propriedades

Os arquivos de propriedades do servidor contêm configurações para executar os servidores de catálogo e servidores de contêiner. É possível especificar um arquivo de propriedades do servidor para uma configuração do WebSphere Application Server ou independente. Os arquivos de propriedades do cliente contêm configurações para o cliente.

Amostras de Arquivos de Propriedades

É possível usar as seguintes amostras de arquivos de propriedades que estão no diretório *extremescale_root*\properties para criar o arquivo de propriedades:

- sampleServer.properties
- sampleClient.properties

Propriedades de Sistema Reprovadas

-Dcom.ibm.websphere.objectgrid.CatalogServerProperties
A propriedade foi reprovada no WebSphere eXtreme Scale Versão 7.0. Use a propriedade **-Dobjectgrid.server.props**.

-Dcom.ibm.websphere.objectgrid.ClientProperties
A propriedade foi reprovada no WebSphere eXtreme Scale Versão 7.0. Use a propriedade **-Dobjectgrid.client.props**.

-Dobjectgrid.security.server.prop
A propriedade foi reprovada no WebSphere eXtreme Scale Versão 6.1.0.3. Use a propriedade **-Dobjectgrid.server.prop**.

-serverSecurityFile
Este argumento foi reprovado no WebSphere eXtreme Scale Versão 6.1.0.3. Essa opção é transmitida no script `startOgServer`. Use a propriedade **-serverProps**.

Arquivo de Propriedades do Servidor

O arquivo de propriedades do servidor contém várias propriedades que definem configurações diferentes para o servidor, como configurações de rastreio, criação de log e configuração de segurança. O arquivo de propriedades do servidor é usado pelo serviço de catálogo e pelos servidores de contêiner.

Amostra do Arquivo de Propriedades do Servidor

É possível usar o arquivo `sampleServer.properties` que esteja no diretório *extremescale_root*`/properties` para criar seu arquivo de propriedades.

Especificando um Arquivo de Propriedades do Servidor

É possível especificar o arquivo de propriedades do servidor de uma das seguintes formas. Especificar uma configuração ao usar um dos itens recentes na lista substitui a configuração anterior. Por exemplo, se você especificar um valor de propriedade de sistema para o arquivo de propriedades do servidor, as propriedades nesse arquivo substituirão os valores no arquivo `objectGridServer.properties` que estiver no caminho de classe.

1. Como um arquivo nomeado corretamente no caminho de classe. Se você colocar esse arquivo nomeado corretamente no diretório atual, o arquivo será localizado apenas se o diretório atual estiver no caminho de classe. O nome usado é o seguinte:

   ```
   objectGridServer.properties
   ```

2. Como uma propriedade do sistema em uma configuração do WebSphere Application Server ou independente que especifica um arquivo no diretório atual do sistema. O arquivo não pode estar no caminho de classe:

   ```
   -Dobjectgrid.server.props=file_name
   ```

3. Como um parâmetro ao executar o comando startOgServer. É possível substituir essas propriedades manualmente para especificar um arquivo no diretório atual do sistema:

   ```
   -serverProps file_name
   ```

4. Como uma substituição programática usando os métodos ServerFactory.getCatalogServerProperties e ServerFactory.getCatalogServerProperties. Os dados no objeto são preenchidos com os dados a partir dos arquivos de propriedades.

Propriedades do Servidor

Propriedades Gerais

workingDirectory
Especifica o local para onde a saída do servidor de contêiner é gravada. Quando esse valor não é especificado, a saída será gravada em um diretório `log` dentro do diretório atual. Essa propriedade se aplica ao servidor de contêiner e ao serviço de catálogo.

Padrão: Nenhum valor

traceSpec
Ativa o rastreio e a cadeia de especificação de rastreio para o servidor de contêiner. O rastreio é desativado por padrão. Essa propriedade se aplica ao servidor de contêiner e ao serviço de catálogo.

Padrão: `*=all=disabled`

traceFile
Especifica um nome de arquivo para gravar informações de rastreio. Essa propriedade se aplica ao servidor de contêiner e ao serviço de catálogo.

systemStreamToFileEnabled
Permite que o contêiner grave SystemOut, SystemErr e a saída de rastreio em um arquivo. Se a propriedade for configurada para `false`, a saída não será gravada em um arquivo e será gravada no console.

Padrão: `true`

enableMBeans
Ativa os beans gerenciados (MBeans) do contêiner ObjectGrid. Essa propriedade se aplica ao servidor de contêiner e ao serviço de catálogo.

Padrão: `true`

serverName
Configura o nome do servidor que é usado para identificar o servidor. Essa propriedade se aplica ao servidor de contêiner e ao serviço de catálogo.

zoneName
Configura o nome da zona à qual o servidor pertence. Essa propriedade se aplica ao servidor de contêiner e ao serviço de catálogo.

HAManagerPort
Especifica o número da porta usada pelo gerenciador de alta disponibilidade. Se essa propriedade não for configurada, o serviço de catálogo gera uma porta disponível automaticamente. Essa propriedade se aplica ao servidor de contêiner e ao serviço de catálogo.

listenerHost
Especifica o nome do host para o qual o Object Request Broker (ORB) deve conectar-se. Essa propriedade se aplica ao servidor de contêiner e ao serviço de catálogo.

listenerPort
Especifica o número da porta para a qual o Object Request Broker (ORB) deve conectar-se. Essa propriedade se aplica ao servidor de contêiner e ao serviço de catálogo.

JMXServicePort
Especifica o número da porta na qual o servidor MBean deve atender. Essa propriedade se aplica ao servidor de contêiner e ao serviço de catálogo.

Propriedades do Servidor de Contêiner

statsSpec
Define a especificação de estatísticas para o servidor de contêiner.

Exemplo:

```
all=disabled
```

memoryThresholdPercentage
Configura o limite de memória para despejo baseado em memória. A porcentagem especifica o heap máximo que deve ser usado no Java Virtual Machine (JVM) antes de a liberação ocorrer. O valor-padrão é `-1`, que indica que o limite de memória não está configurado. Se a propriedade memoryThresholdPercentage não for configurada, o valor MemoryPoolMXBean será configurado com o valor fornecido. Consulte Interface MemoryPoolMXBean na especificação de API Java para obter mais informações. Porém, a liberação ocorre apenas se ela for ativada em um evictor. Para ativar a liberação baseada em memória, consulte as informações sobre os evictors no *Visão Geral do Produto*. Essa propriedade é aplicada apenas a um servidor de contêiner.

catalogServiceEndPoints
Especifica os terminais para conexão com o cluster do serviço de catálogo. Este valor deve estar no formato `host:port<,host:port>` em que o valor do host é o valor listenerHost e o valor da porta é o valor listenerPort do servidor de catálogo. Essa propriedade é aplicada apenas a um servidor de contêiner.

Propriedades do Serviço de Catálogo

domainName
Especifica o nome do domínio que é usado para identificar exclusivamente essa grade de serviço de catálogo para os clientes ao serem roteados para vários domínios. Essa propriedade é aplicada apenas no serviço de catálogo.

enableQuorum
Ativa o quorum para o serviço de catálogo. O quorum é usado para garantir que a maioria da grade do serviço de catálogo esteja disponível antes de permitir a modificação da colocação das partições nos servidores de contêineres disponíveis. Para ativar o quorum, configure o valor para `true` ou `ativado`. O valor padrão é `disabled`. Essa propriedade é aplicada apenas no serviço de catálogo.

catalogClusterEndpoints
Especifica os terminais da grade do serviço de catálogo para o serviço de catálogo. Essa propriedade especifica os terminais do serviço de catálogo para iniciar a grade do serviço de catálogo. Utilize o seguinte formato:

```
serverName:hostName:clientPort:peerPort<serverName:hostName:clientPort:peerPort>
```

Essa propriedade é aplicada apenas no serviço de catálogo.

heartBeatFrequencyLevel
Especifica com que frequência a pulsação ocorre. O nível de frequência de pulsação é uma troca entre o uso dos recursos e a hora da descoberta de falha. Quanto maior for a frequência das pulsações, mais recursos serão usados e as falhas serão descobertas mais rapidamente. Essa propriedade é aplicada apenas no serviço de catálogo. Utilize um dos seguintes valores:

- `0`: Especifica o nível de pulsação a uma taxa normal. Com esse valor, a detecção de failover ocorrerá a uma taxa razoável sem usar excessivamente os recursos. (Default)
- `-1`: Especifica um nível de pulsação forte. Com esse valor, as falhas serão detectadas mais rapidamente, mas também são usados um processador e recursos de rede adicionais. Esse nível pode ter pulsações ausentes quando o servidor estiver ocupado.
- 1: Especifica um nível de pulsação livre. Com esse valor, uma frequência de pulsação diminuída aumenta o tempo para detectar falhas, mas também diminui o uso do processador e de rede.

Propriedades do Servidor de Segurança

O arquivo de propriedades do servidor também é usado para configurar a segurança do servidor eXtreme Scale. Use um arquivo de propriedades único do servidor para especificar as propriedades básicas e de segurança.

Propriedades gerais de segurança

securityEnabled
Ativa a segurança do servidor de contêiner ao configurar para `true`. O valor padrão é `false`. Esta propriedade deve corresponder à propriedade securityEnabled que é especificada no arquivo `objectGridSecurity.xml` que é fornecido para o servidor de catálogos.

credentialAuthentication
Indica se este servidor suporta a autenticação de credencial. Escolha um dos seguintes valores:

- `Nunca`: O servidor não suporta a autenticação de credencial.
- `Suportado`: O servidor suporta a autenticação de credencial se o cliente também suportar a autenticação de credencial.
- `Necessário`: O cliente requer autenticação de credencial.

Consulte o "Autenticação de Cliente do Aplicativo" na página 287 para obter detalhes sobre a autenticação de credencial.

Configurações de segurança da camada de transporte

transportType
Especifica o tipo de transporte do servidor. Utilize um dos seguintes valores:

- `TCP/IP`: Indica que o servidor suporta apenas conexões TCP/IP.
- `SSL-Suportado`: Indica que o servidor suporta ambas as conexões TCP/IP e Secure Sockets Layer (SSL). (Default)
- `SSL-Necessário`: Indica que o servidor requer as conexões SSL.

Propriedades de Configuração SSL

alias Especifica o nome do alias no armazenamento de chaves. Esta propriedade será utilizada se o armazenamento de chaves tiver vários certificados de pares de chaves e você desejar selecionar um dos certificados.

Padrão: nenhum valor

contextProvider
Especifica o nome do provedor de contexto para o serviço de confiança. Se você indicar um valor inválido, ocorrerá uma exceção de segurança que indica que o tipo de provedor de contexto está incorreto.

Valores válidos: `IBMJSSE2`, `IBMJSSE`, `IBMJSSEFIPS`, etc.

protocol
Indica o tipo de protocolo de segurança a ser usado para o cliente. Configure esse valor de protocolo baseado no provedor Java Secure Socket Extension (JSSE) que será usado. Se você indicar um valor inválido, ocorrerá uma exceção de segurança que indica que o valor de protocolo está incorreto.

Valores válidos: `SSL`, `SSLv2`, `SSLv3`, `TLS`, `TLSv1`, etc.

keyStoreType
Indica o tipo de armazenamento de chaves. Se você indicar um valor inválido, ocorrerá uma exceção de segurança no tempo de execução.

Valores válidos: `JKS`, `JCEK`, `PKCS12`, etc.

trustStoreType
Indica o tipo de armazenamento de confiança. Se você indicar um valor inválido, ocorrerá uma exceção de segurança no tempo de execução.

Valores válidos: `JKS`, `JCEK`, `PKCS12`, etc.

keyStore
Especifica um caminho completo para o arquivo de armazenamento de chaves.

Exemplo:

```
etc/test/security/client.private
```

trustStore
Especifica um caminho completo para o arquivo de armazenamento de confiança.

Exemplo:

```
etc/test/security/server.public
```

keyStorePassword
Especifica a senha da cadeia para o armazenamento de chaves. É possível codificar este valor ou usar o valor real.

trustStorePassword
Especifica uma senha de cadeia para o armazenamento de confiança. É possível codificar este valor ou usar o valor real.

clientAuthentication
Se a propriedade for configurada para true, o cliente SSL deverá ser autenticado. A autenticação do cliente SSL é diferente da autenticação do certificado cliente. A autenticação do certificado cliente significa autenticar um cliente para um registro de usuário de acordo com a cadeia de certificados. Essa propriedade garante que o servidor se conecte ao cliente correto.

Configuração SecureTokenManager

A configuração SecureTokenManager é usada para proteger a cadeia secreta para autenticações mútuas do servidor e para proteger o token de conexão única.“Segurança de Grade” na página 286

secureTokenManagerType
Especifica o tipo de configuração SecureTokenManager. É possível usar uma das seguintes configurações:

- `nenhum`: Indica que nenhum gerenciador de token seguro é usado.

- `padrão`: Indica que o gerenciador de token que é fornecido com o produto WebSphere eXtreme Scale é usado. É necessário fornecer uma configuração de armazenamento de chaves SecureToken.
- `custom`: Indica que você tem seu próprio gerenciador de token que você especificou com a classe de implementação do SecureTokenManager.

customTokenManagerClass
Especifica o nome da sua classe de implementação SecureTokenManager, se você tiver especificado o valor de propriedade SecureTokenManagerType como `customizado`. A classe de implementação deve ter um construtor padrão a ser instanciado.

customSecureTokenManagerProps
Especifica as propriedades de classe de implementação SecureTokenManager customizada. Essa propriedade é usada apenas se o valor secureTokenManagerType for `customizado`. O valor é configurado para o objeto SecureTokenManager com o método setProperties(String).

Configuração de armazenamento de chaves de token seguro

secureTokenKeyStore
Especifica o nome do caminho de arquivo para o armazenamento de chaves que armazena o par de chaves público-privado e a chave secreta.

secureTokenKeyStoreType
Especifica o tipo de armazenamento de chaves, por exemplo, JCKES. É possível configurar esse valor baseado no provedor Java Secure Socket Extension (JSSE) usado. No entanto, esse armazenamento de chaves deve suportar chaves secretas.

secureTokenKeyPairAlias
Especifica o alias do par de chaves público-privado que é usado para assinatura e verificação.

secureTokenKeyPairPassword
Especifica a senha para proteger o alias de par de chaves que é usado para assinatura e verificação.

secureTokenSecretKeyAlias
Especifica o alias de chave secreta que é usado para codificação.

secureTokenSecretKeyPassword
Especifica a senha para proteger a chave secreta.

secureTokenCipherAlgorithm
Especifica o algoritmo que é usado para fornecer uma codificação. É possível configurar esse valor baseado no provedor Java Secure Socket Extension (JSSE) usado.

secureTokenSignAlgorithm
Especifica o algoritmo que é usado para assinar o objeto. Você pode configurar esse valor com base no provedor JSSE utilizado.

Cadeia de autenticação

authenticationSecret
Especifica a cadeia secreta para fazer o pedido ao servidor. Quando um servidor for inicializado, ele deverá apresentar essa cadeia ao servidor principal ou ao servidor de catálogos. Se a cadeia secreta corresponder com o que está no servidor principal, esse servidor poderá fazer a união.

Arquivo de Propriedades do Cliente

É possível criar um arquivo de propriedades com base nos requisitos para os processos do cliente do eXtreme Scale.

Amostra do Arquivo de Propriedades do Cliente

É possível usar o arquivo `sampleClient.properties` que esteja no diretório *extremescale_root*`\properties` para criar seu arquivo de propriedades.

Especificando um Arquivo de Propriedades do Cliente

É possível especificar o arquivo de propriedades do cliente de uma das seguintes formas. Especificar uma configuração ao usar um dos itens recentes na lista substitui a configuração anterior. Por exemplo, se você especificar um valor de propriedade de sistema para o arquivo de propriedades do cliente, as propriedades nesse arquivo substituirão os valores no arquivo `objectGridClient.properties` que estiver no caminho de classe.

1. Como um arquivo nomeado corretamente em qualquer lugar no caminho de classe. Colocar esse arquivo no diretório atual do sistema não é suportado:
   `objectGridClient.properties`
2. Como uma propriedade do sistema em uma configuração do WebSphere Application Server ou independente. Este valor pode especificar um arquivo no diretório atual do sistema, mas não um arquivo no caminho de classe:
   `-Dobjectgrid.client.props=`*`file_name`*
3. Como uma substituição programática usando o método ClientClusterContext.getClientProperties. Os dados no objeto são preenchidos com os dados a partir dos arquivos de propriedades. Não é possível configurar as propriedades de segurança com esse método.

Propriedades do Cliente

preferLocalProcess
: Especifica se o processo local é preferido para roteamento. Quando configurado para true, os pedidos são roteados para os shards que são colocados no mesmo processo do cliente quando apropriado.

  Padrão: `true`

preferLocalHost
: Especifica se o host local é preferido para roteamento. Quando configurado para true, os pedidos são roteados para os shards que são colocados no mesmo host do cliente quando apropriado.

  Padrão: `true`

preferZones
: Especifica uma lista de zonas de roteamento preferidas. Cada zona especificada é separada por vírgula no formato:
  `preferZones=ZoneA,ZoneB,ZoneC`

  Padrão: Nenhum valor

requestRetryTimeout
: Especifica quanto tempo um pedido será tentado novamente (em milissegundos). Utilize um dos seguintes valores válidos:
  - O valor `0` indica que o pedido deve falhar rapidamente e ignorar a lógica de nova tentativa interna.

- O valor `-1` indica que o tempo limite da nova tentativa do pedido não foi configurado, significando que a duração do pedido é controlada pelo tempo limite da transação. (Default)
- O valor `0` indica o valor do tempo limite de entrada do pedido em milissegundos. As exceções que não puderem ocorrer mesmo se for tentado novamente como uma exceção DuplicateException serão retornadas imediatamente. O tempo limite da transação ainda é usado como o tempo máximo de espera.

Propriedades de Segurança do Cliente

Propriedades gerais de segurança

securityEnabled
Ativa a segurança do cliente do WebSphere eXtreme Scale. Essa configuração de segurança ativada deve corresponder com a configuração securityEnabled no arquivo de propriedades do servidor WebSphere eXtreme Scale. Se as configurações não corresponderem, uma exceção ocorrerá.

Padrão: `false`

Propriedades de Configuração de Autenticação de Credencial

credentialAuthentication
Especifica o suporte de autenticação da credencial do cliente. Utilize um dos seguintes valores válidos:

- `Nunca`: O cliente não suporta a autenticação de credencial.
- `Suportado`: O cliente suporta a autenticação de credencial se o servidor também suportar a autenticação de credencial. (Default)
- `Necessário`: O cliente requer autenticação de credencial.

authenticationRetryCount
Especifica o número de vezes em que a autenticação é tentada novamente se a credencial expirar. Se o valor for configurado para 0, as tentativas de autenticação não serão feitas novamente.

Padrão: `3`

credentialGeneratorClass
Especifica o nome da classe que implementa a interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. Essa classe é usada para obter credenciais para os clientes.

Padrão: Nenhum valor

credentialGeneratorProps
Especifica as propriedades para a classe de implementação CredentialGenerator. As propriedades são configuradas para o objeto com o método setProperties(String). O valor credentialGeneratorprops é usado apenas se o valor da propriedade credentialGeneratorClass não for nulo.

Propriedades da configuração de segurança da camada de transporte

transportType
Especifica o tipo de transporte do cliente. Os valores possíveis são:

- `TCP/IP`: Indica que o cliente suporta apenas conexões TCP/IP.
- `SSL-Suportado`: Indica que o cliente suporta ambas as conexões TCP/IP e Secure Sockets Layer (SSL). (Default)
- `SSL-Necessário`: Indica que o cliente requer as conexões SSL.

Propriedades de Configuração SSL

alias Especifica o nome do alias no armazenamento de chaves. Esta propriedade será utilizada se o armazenamento de chaves tiver vários certificados de pares de chaves e você desejar selecionar um dos certificados.

Padrão: nenhum valor

contextProvider
Especifica o nome do provedor de contexto para o serviço de confiança. Se você indicar um valor inválido, ocorrerá uma exceção de segurança que indica que o tipo de provedor de contexto está incorreto.

Valores válidos: `IBMJSSE2`, `IBMJSSE`, `IBMJSSEFIPS`, etc.

protocol
Indica o tipo de protocolo de segurança a ser usado para o cliente. Configure esse valor de protocolo baseado no provedor Java Secure Socket Extension (JSSE) que será usado. Se você indicar um valor inválido, ocorrerá uma exceção de segurança que indica que o valor de protocolo está incorreto.

Valores válidos: `SSL`, `SSLv2`, `SSLv3`, `TLS`, `TLSv1`, etc.

keyStoreType
Indica o tipo de armazenamento de chaves. Se você indicar um valor inválido, ocorrerá uma exceção de segurança no tempo de execução.

Valores válidos: `JKS`, `JCEK`, `PKCS12`, etc.

trustStoreType
Indica o tipo de armazenamento de confiança. Se você indicar um valor inválido, ocorrerá uma exceção de segurança no tempo de execução.

Valores válidos: `JKS`, `JCEK`, `PKCS12`, etc.

keyStore
Especifica um caminho completo para o arquivo de armazenamento de chaves.

Exemplo:

```
etc/test/security/client.private
```

trustStore
Especifica um caminho completo para o arquivo de armazenamento de confiança.

Exemplo:

```
etc/test/security/server.public
```

keyStorePassword
Especifica a senha da cadeia para o armazenamento de chaves. É possível codificar este valor ou usar o valor real.

trustStorePassword
Especifica uma senha de cadeia para o armazenamento de confiança. É possível codificar este valor ou usar o valor real.

Arquivo de Propriedades ORB

O arquivo `orb.properties` é usado para transmitir as propriedades que são usadas pelo Object Request Broker (ORB) para modificar o comportamento de transporte da grade.

Local

O arquivo `orb.properties` está no diretório `java/jre/lib`. Ao modificar o arquivo em um diretório WebSphere Application Server `java/jre/lib`, os servidores de aplicativos que são configurados nessa instalação também usam as configurações do arquivo.

Configurações de Linha de Base

As seguintes configurações são uma linha de base ideal mas não necessariamente as melhores configurações para cada ambiente. É necessário entender as configurações para ajudar a tomar uma boa decisão sobre quais valores são apropriados no seu ambiente.

```
com.ibm.CORBA.RequestTimeout=30
com.ibm.CORBA.ConnectTimeout=10
com.ibm.CORBA.FragmentTimeout=30
com.ibm.CORBA.ThreadPool.MinimumSize=256
com.ibm.CORBA.ThreadPool.MaximumSize=256
com.ibm.CORBA.ThreadPool.IsGrowable=false
com.ibm.CORBA.ConnectionMultiplicity=1
com.ibm.CORBA.MinOpenConnections=1024
com.ibm.CORBA.MaxOpenConnections=1024
com.ibm.CORBA.ServerSocketQueueDepth=1024
com.ibm.CORBA.FragmentSize=0
com.ibm.CORBA.iiop.NoLocalCopies=true
com.ibm.CORBA.NoLocalInterceptors=true
```

Descrições de Propriedades

Configurações de Tempo Limite

As seguintes configurações referem-se à quantia de tempo em que o ORB aguardará antes que as operações sobre pedidos sejam canceladas.

Tempo Limite do Pedido

Nome da Propriedade: com.ibm.CORBA.RequestTimeout

Valor: Valor de número inteiro para número de segundos.

Descrição: Indica quantos segundos um pedido deve aguardar por uma resposta antes de ser cancelado. Essa propriedade influencia a quantia de tempo em que um cliente precisa para efetuar failover se ocorrer uma falha de interrupção de rede. Se você configurar essa propriedade muito baixo, os pedidos poderão atingir o tempo limite involuntariamente. Considere cuidadosamente o valor dessa propriedade para evitar tempos limites involuntários.

Tempo Limite de Conexão

Nome da Propriedade: com.ibm.CORBA.ConnectTimeout

Valor: Valor de número inteiro para número de segundos.

Descrição: Indica quantos segundos uma tentativa de conexão de soquete deve aguardar antes de ser cancelada. Essa propriedade, como o tempo limite de pedido, pode influenciar no tempo em que um cliente precisa para efetuar failover se ocorrer uma falha de interrupção de rede. Em geral, configure essa propriedade para um valor menor do que o valor do tempo limite de pedido porque a quantia de tempo para estabelecer uma conexão deve ser relativamente constante.

Tempo Limite de Fragmento

Nome da Propriedade: com.ibm.CORBA.FragmentTimeout

Valor: Valor de número inteiro para número de segundos.

Descrição: Indica quantos segundos um pedido de fragmento deve aguardar antes de ser cancelado. Essa propriedade é semelhante à propriedade de tempo limite de pedido.

Configurações do Conjunto de Encadeamentos

Essas propriedades restringem o tamanho do conjunto de encadeamentos a um número específico de encadeamentos. Os encadeamentos são usados pelo ORB para distribuir os pedidos do servidor depois de serem recebidos no soquete. Configurar esses valores da propriedade muito baixo resulta em um aumento de profundidade da fila de soquete e possivelmente a ocorrência de tempos limites.

Multiplicidade de Conexão

Nome da Propriedade: com.ibm.CORBA.ConnectionMultiplicity

Valor: Valor de número inteiro para número de conexões entre o cliente e o servidor. O valor padrão é 1. Configurar um valor maior define uma multiplicidade entre várias conexões.**Descrição:** Permite que o ORB use várias conexões em qualquer servidor. Na teoria, configurar esse valor deve promover paralelismo sobre as conexões. Na prática, o desempenho não é beneficiado pela configuração da multiplicidade de conexão. Não configure esse parâmetro.

Conexões Abertas

Nomes da Propriedade: com.ibm.CORBA.MinOpenConnections, com.ibm.CORBA.MaxOpenConnections

Valor: Valor de número inteiro para número de conexões.**Descrição:** Especifica um número mínimo e máximo de conexões abertas. O ORB mantém um cache de conexões que foram estabelecidas com os clientes. Essas conexões são limpas quando o valor de com.ibm.CORBA.MaxOpenConnections for transmitido. A limpeza das conexões pode prejudicar o comportamento na grade.

É Aumentável

Nome da Propriedade: com.ibm.CORBA.ThreadPool.IsGrowable

Valor: Booleano; configurado para `true` ou `false`.**Descrição:** Se ativado, permite que o conjunto de encadeamentos que o ORB utiliza para pedidos recebidos aumente além do que o conjunto suporta. Se o tamanho do conjunto for excedido, novos encadeamentos serão criados para manipular o pedido, mas os encadeamentos não serão agrupados.

Profundidade da Fila de Soquete do Servidor

Nome da Propriedade: com.ibm.CORBA.ServerSocketQueueDepth

Valor: Valor de número inteiro para número de conexões.**Descrição:** Especifica o comprimento da fila para conexões recebidas dos clientes. As filas ORB recebem conexões dos clientes. Se a fila estiver cheia, as conexões serão recusadas. Recusar conexões pode prejudicar o comportamento na grade.

Tamanho do Fragmento

Nome da Propriedade: com.ibm.CORBA.FragmentSize

Valor: Um número inteiro que especifica o número de bytes. O padrão é 1024.**Descrição:** Especifica o tamanho máximo do pacote que o ORB usa ao enviar um pedido. Se um pedido for maior que o limite de tamanho de fragmento, esse pedido será dividido em fragmentos de pedido e enviados, cada um, separadamente e remontados no servidor. Os pedidos de fragmentação são úteis em redes não-confiáveis onde os pacotes podem precisar ser reenviados. Porém, se a rede for confiável, dividir os pedidos em fragmentos pode causar sobrecarga.

Nenhuma Cópia Local

Nome da Propriedade: com.ibm.CORBA.iiop.NoLocalCopies

Valor: Booleano; configurado para `true` ou `false`. **Descrição:** Especifica se o ORB é transmitido por referência. O ORB usa a chamada transmitir por valor por padrão. A chamada transmitir por valor causa um custo de serialização extra desnecessário no caminho quando uma interface for chamada localmente. Ao configurar esse valor para true, o ORB usa um método transmitir por referência que é mais eficiente do que a chamada transmitir por valor.

Nenhum Interceptor Local

Nome da Propriedade: com.ibm.CORBA.NoLocalInterceptors

Valor: Booleano; configurado para `true` ou `false`. **Descrição:** Especifica se o ORB chama os interceptores de pedido mesmo ao fazer pedidos locais (intraprocesso). Os interceptores que o WebSphere eXtreme Scale usa são para manipulação de segurança e de rota, e não são necessários se o pedido manipular o processo no qual ele está em execução. Os interceptores que estão entre os processos são necessários apenas para as operações de chamada de procedimento remoto (RPC). Ao configurar 'Nenhum interceptor local', você pode evitar a sobrecarga extra ao usar os interceptores locais.

Quando quiser aplicar segurança de transporte entre os clientes e servidores do ObjectGrid, é necessário incluir mais propriedades no arquivo `orb.properties`. Para obter mais informações sobre essas propriedades, consulte a seção sobre o arquivo `orb.properties` para suporte de segurança de transporte no "Transport Layer Security e Secure Sockets Layer" na página 293.

Integração com a Estrutura Spring

O Spring é uma estrutura popular para desenvolvimento de aplicativos Java. O WebSphere eXtreme Scale fornece suporte para permitir que o Spring gerencie as transações do eXtreme Scale e configure clientes e servidores que compõem a grade de dados de memória implementada.

Transações Nativas Gerenciadas do Spring

O Spring fornece transações gerenciadas por contêiner que são similares a um servidor de aplicativos do Java Platform, Enterprise Edition. Porém, o mecanismo do Spring pode se conectar em diferentes implementações. O WebSphere eXtreme Scale fornece a integração do gerenciador de transações que permite ao Spring para gerenciar os ciclos de vida da transação do ObjectGrid. Consulte as informações sobre transações nativas no *Guia de Programação* para obter detalhes.

Beans de Extensão Gerenciados do Spring e Suporte a Espaço de Nomes

Além disso, o eXtreme Scale se integra ao Spring para permitir que os beans de estilo do Spring definidos para pontos de extensão ou plug-ins. Este recurso fornece configurações mais sofisticadas e mais flexíveis para configuração dos pontos de extensão.

Além dos beans de extensão gerenciados do Spring, o eXtreme Scale fornece um espaço de nomes Spring chamado ″objectgrid″. Beans e implementações integradas são predefinidos neste espaço de nomes, o que facilita aos usuários configurar o eXtreme Scale. Consulte o “Beans de Extensão Spring e Suporte a Espaço de Nomes” para obter detalhes adicionais sobre este tópicos e um exemplo sobre como iniciar um servidor de contêineres do eXtreme Scale usando as configurações do Spring.

Suporte ao Escopo Shard

Com a configuração do Spring estilo tradicional, um bean ObjectGrid pode se do tipo singleton ou prototype. O ObjectGrid também suporta um novo escopo chamado de escopo ″shard″. Se um bean for definido como escopo shard, então somente um bean será criado por shard. Todos os pedidos para beans com um ID ou IDs correspondentes a esta definição de bean no mesmo shard resultarão naquela instância específica do bean retornada pelo contêiner do Spring.

O exemplo a seguir mostra que um bean com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl está definido com escopo definido como shard. Desta forma, somente uma instância da classe JPAPropFactoryImpl é criada por shard.

```
<bean id="jpaPropFactory" class="com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl" scope="shard" />
```

Fluxo da Web do Spring

O Fluxo da Web do Spring armazena o estado da sua sessão na Sessão HTTP pelo padrão. Se um aplicativo da Web for configurado para usar o eXtreme Scale para o gerenciamento de sessão, então ele será usado automaticamente pelo Spring para armazenar este estado e se tornará tolerante a falhas da mesma forma que a sessão.

compactando

As extensões Spring do eXtreme Scale estão no arquivo `ogspring.jar`. Este arquivo Java archive (JAR) deve estar no caminho de classe para o suporte ao Spring funcionar. Se um aplicativo JEE que está em execução em um WebSphere Extended Deployment alterado WebSphere Application Server Network Deployment, então o aplicativo deve colocar o arquivo `spring.jar` e seus arquivos associados nos módulos EAR (Enterprise Archive). Você também deve colocar o arquivo `ogspring.jar` no mesmo local.

Beans de Extensão Spring e Suporte a Espaço de Nomes

O WebSphere eXtreme Scale fornece um recurso para declarar POJO (Plain Old Java Objects) para usar como pontos de extensão no arquivo `objectgrid.xml` e uma forma de nomear os beans e, em seguida, especificar o nome da classe. Normalmente, as instâncias da classe especificada são criadas, e tais objetos são usados como plug-ins. Agora, o eXtreme Scale pode delegar que Spring obtenha

instâncias destes objetos plug-in. Se um aplicativo utiliza o Spring, então, normalmente, tais POJOs possuem um requisito de serem conectados ao resto do aplicativo.

Em alguns casos, você deve usar o Spring para configurar determinados objetos do plug-in. Use a configuração a seguir como exemplo:

```
<objectGrid name="Grid">
    <bean id="TransactionCallback" className="com.ibm.websphere.objectgrid.jpa.JPATxCallback">
        <property name="persistenceUnitName" type="java.lang.String"  value="employeePU" />
    </bean>
    ...
</objectGrid>
```

A implementação de TransactionCallback integrada, a classe com.ibm.websphere.objectgrid.jpa.JPATxCallback é configurada como a classe TransactionCallback. Este classe é configurada como uma propriedade persistenceUnitName conforme mostrado no exemplo anterior. A classe JPATxCallback também possui um atributo JPAPropertyFactory, que é o tipo java.lang.Object. A configuração XML do ObjectGrid não pode suportar este tipo de configuração.

A integração Spring do eXtreme Scale soluciona este problema delegando a criação do bean à estrutura do Spring. A configuração revisada é a seguinte:

```
<objectGrid name="Grid">
    <bean id="TransactionCallback" className="{spring}jpaTxCallback"/>
    ...
</objectGrid>
```

O arquivo Spring para o objeto ″Grid″ contém as seguintes informações:

```
<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>
```

Aqui, o TransactionCallback está especificado como `{spring}jpaTxCallback`, e os beans jpaTxCallback e jpaPropFactory estão configurados no arquivo Spring como mostrado no exemplo anterior. A configuração Spring torna possível a configuração de um bena JPAPropertyFactory como um parâmetro do objeto JPATxCallback.

Spring bean factory padrão

Quando o eXtreme Scale encontra um plug-in ou um bean de extensão (como um ObjectTransformer, utilitário de carga, TransactionCallback e assim por diante) com um valor className que inicia com o prefixo `{spring}`, o eXtreme Scale usará o restante do nome como um nome Spring Bean e obterá a instância do bean usando o Spring Bean Factory.

Pelo padrão, se nenhum bean factory tiver sido registrado para um determinado ObjectGrid, então ele tenta localizar um arquivo `ObjectGridName_spring.xml`. Por exemplo, se sua grade for chamada como ″Grid″, então o arquivo XML será chamado como `/Grid_spring.xml`. Este arquivo deve estar no caminho da classe ou em um diretório `META-INF` que está no caminho da classe. Se este arquivo for encontrado, então o eXtreme Scale construi um ApplicationContext usando tal arquivo e construi beans a partir desse bean factory.

Spring bean factory customizado

O WebSphere eXtreme Scale também fornece uma API ObjectGridSpringFactory para registrar uma instância do Spring Bean Factory para usar para um ObjectGrid específico nomeado. Esta API registra uma instância de BeanFactory com eXtreme Scale ussando o método estático a seguir:

```
void registerSpringBeanFactoryAdapter(String objectGridName, Object
springBeanFactory)
```

Suporte a Espaço de Nomes

Desde a versão 2.0, o Spring possui um mecanismo para extensão baseado em esquema para o formato XML do Spring básico para definição e configuração de beans. O ObjectGrid usa este novo recurso para definir e configurar beans ObjectGrid. Com a extensão de esquema XML do Spring, algumas das implementações integradas dos plug-ins do eXtreme Scale e alguns beans do ObjectGrid são predefinidos no espaço de nomes ″objectgrid″. Ao escrever arquivos de configuração Spring, você não tem que especificar o nome de classe integral dos integrados. Em vez disso, é possível referenciar os beans predefinidos.

Além disso, com os atributos dos beans definidos no esquema XML, é menos provável que você forneça um nome de atributo errado. A validação XML baseada no esquema XML pode capturar estes tipos de erros anteriormente no ciclo de desenvolvimento.

Estes beans definidos nas extensões de esquema XML são:

- transactionManager
- register
- server
- catálogo
- contêiner
- JPALoader
- JPATxCallback
- JPAEntityLoader
- LRUEvictor
- LFUEvictor
- HashIndex

Estes beans são definidos no esquema XML objectgrid.xsd. Este arquivo XSD é enviado como arquivo `com/ibm/ws/objectgrid/spring/namespace/objectgrid.xsd` no arquivo `ogspring.jar`. Para obter descrições detalhadas do arquivo XSD e dos beans definidos no arquivo XSD, consulte as informações sobre o arquivo descritor Spring no *Guia de Administração*.

Continue usando o exemplo JPATxCallback da seção anterior. Na seção anterior, o bean JPATxCallback é configurado como o seguinte:

```
<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>
```

Usando este recurso de espaço de nomes, a configuração XML do Spring pode ser escrita da seguinte forma:

```
<objectgrid:JPATxCallback id="jpaTxCallback" persistenceUnitName="employeeEMPU"
 jpaPropertyFactory="jpaPropFactory" />

<bean id="jpaPropFactory" class="com.ibm.ws.objectgrid.jpa.plugins.JPAPropFactoryImpl"
 scope="shard">
</bean>
```

Observe aque que em vez de especificar a classe ″com.ibm.websphere.objectgrid.jpa.JPATxCallback″ como no exemplo anterior, foi usado diretamente o bean ″objectgrid:JPATxCallback″ predefinido. Como pode ser visto, esta configuração é menos detalhada e mais amigável para verificação de erro.

Início com Servidor de Contêineres com Spring Extension Beans

Neste exemplo, será mostrado como iniciar um servidor ObjectGrid usando beans de extensão gerenciados Spring do ObjectGrid e suporte a espaço de nomes.

Arquivo XML do ObjectGrid

Primeiramente, defina um arquivo XML do ObjectGrid muito simples que contenha um ″Grid″ do ObjectGrid e uma mapa ″Test″. O ObjectGrid possui um plug-in ObjectGridEventListener chamado ″partitionListener″, e o mapa ″Test″ possui um Evictor conectado chamado ″testLRUEvictor″. Observe que ambos os plug-ins ObjectGridEventListener e Evictor são configurados usando Spring pois seus nomes contêm ″{spring}″.

```
<?xml version="1.0" encoding="UTF-8"?>
<objectGridConfig xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://ibm.com/ws/objectgrid/config ../objectGrid.xsd"
 xmlns="http://ibm.com/ws/objectgrid/config">
    <objectGrids>
        <objectGrid name="Grid">
     <bean id="ObjectGridEventListener" className="{spring}partitionListener" />
            <backingMap name="Test" pluginCollectionRef="test" />
        </objectGrid>
    </objectGrids>

    <backingMapPluginCollections>
        <backingMapPluginCollection id="test">
            <bean id="Evictor" className="{spring}testLRUEvictor"/>
        </backingMapPluginCollection>
    </backingMapPluginCollections>
</objectGridConfig>
```

Arquivo XML de implementação do ObjectGrid

Agora, crie um arquivo XML de implementação simples do ObjectGrid da forma a seguir. Ele particiona o ObjectGrid em 5 partições, e nenhuma réplica é necessária.

```
<?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>
```

Arquivo XML Spring do ObjectGrid

Agora serão usados tanto beans de extensão gerenciado Spring do ObjectGrid e recursos de suporte a espaço de nomes para configurar os beans ObjectGrid. O arquivo XML do Spring é nomeado como ″Grid_spring.xml″. Observe que estão incluídos dois esquemas no arquivo XML: spring-beans-2.0.xsd é para uso com benas gerenciados do Spring, e objectgrid.xsd é para uso com beans predefinidos no espaço de nomes objectgrid.

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:objectgrid="http://www.ibm.com/schema/objectgrid"
       xsi:schemaLocation="
       http://www.ibm.com/schema/objectgrid
   http://www.ibm.com/schema/objectgrid/objectgrid.xsd
       http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

    <objectgrid:register id="ogregister" gridname="Grid"/>

    <objectgrid:server id="server" isCatalog="true" name="server">
        <objectgrid:catalog host="localhost" port="2809"/>
    </objectgrid:server>

    <objectgrid:container id="container"
   objectgridxml="com/ibm/ws/objectgrid/test/springshard/objectgrid.xml"
       deploymentxml="com/ibm/ws/objectgrid/test/springshard/deployment.xml"
   server="server"/>

    <objectgrid:LRUEvictor id="testLRUEvictor" numberOfLRUQueues="31"/>

    <bean id="partitionListener"
   class="com.ibm.websphere.objectgrid.springshard.ShardListener" scope="shard"/>
</beans>
```

Há 6 beans definidos neste arquivo XML do Spring:

1. *objectgrid:register*: Isto registra o bean factory padrão para o ″Grid″ do ObjectGrid.
2. *objectgrid:server*: Isto define um servidor do ObjectGrid com o nome ″server″. Este servidor também fornece o serviço de catálogo desde que ele possua um bean objectgrid:catalog aninhado nele.
3. *objectgrid:catalog*: Isto define um terminal de serviço de catálogo ObjectGrid, que está configurado como ″localhost:2809″.
4. *objectgrid:container*: Isto define um contêiner ObjectGrid com o arquivo XML objectgrid especificado e o arquivo XML de implementação como discutido anteriormente. A propriedade de servidor especifica em qual servidor este contêiner está hospedado.
5. *objectgrid:LRUEvictor*: Isto define um LRUEvictor com a quantidade de filas LRU para usar configurada como 31.
6. *bean partitionListener*: Isto define um plug-in ShardListener. Esta classe é uma classe conectada por usuários, assim ela não pode usar beans predefinidos. Este escopo do bean também é configurado como ″shard″, o que significa que há somente uma instância deste ShardListener por shard ObejctGrid.

Início do servidor

O fragmento a seguir inicia o servidor ObjectGrid, que hospeda tanto o serviço de contêiner e o serviço de catálogo. Com pode ser visto, o único método que precisa ser chamado para iniciar o servidor e para obter um bean ″container″ no factory do bean. Isto simplifica a complexidade de programação pela movimentação da maioria da lógica na configuração do Spring.

```
public class ShardServer extends TestCase
{
    Container container;
    org.springframework.beans.factory.BeanFactory bf;

    public void startServer(String cep)
    {
        try
        {
            bf = new org.springframework.context.support.ClassPathXmlApplicationContext(
                "/com/ibm/ws/objectgrid/test/springshard/Grid_spring.xml", ShardServer.class);
            container = (Container)bf.getBean("container");
        }
        catch(Exception e)
        {
            throw new ObjectGridRuntimeException("Cannot start OG container", e);
        }
```

```
    }

    public void stopServer()
    {
        if(container != null)
            container.teardown();
    }
}
```

Arquivo XML descritor do Spring

Use um arquivo XML descritor do Spring para configurar e integrar o eXtreme Scale com o Spring.

Nas seguintes seções, cada elemento e atributo do arquivo `objectgrid.xsd` Spring é definido. O arquivo `objectgrid.xsd` Spring está no arquivo `ogspring.jar` e no espaço de nomes objectgrid `com/ibm/ws/objectgrid/spring/namespace`. Consulte o "Arquivo `objectgrid.xsd` Spring" na página 210 para obter um exemplo do esquema do XML descritor.

Elemento register

Use o elemento register para registrar o factory do bean padrão para a grade ObjectGrid.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

Atributos

id Especifica o nome do diretório bean padrão para um determinado ObjectGrid.

gridname
Especifica o nome da instância do ObjectGrid. O valor designado para esse atributo deve corresponder a um ObjectGrid válido configurado no arquivo descritor ObjectGrid.

```
<register
(1) id="register id"
(2) gridname="ObjectGrid name"
/>
```

Elemento server

Use o elemento server para definir um servidor eXtreme Scale, que pode hospedar um contêiner, um serviço de catálogo, ou ambos.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

Atributos

id Especifica o nome do servidor eXtreme Scale.

tracespec
Indica o tipo de rastreio e permite o rastreio e a especificação de rastreio para o servidor.

tracefile
Fornece o caminho e o nome do traceFile a ser criado e usado.

statspec
Indica a especificação de estatísticas para o servidor.

jmxport
Designa o número de porta não-utilizada através da qual você deseja ativar as conexões JMX/RMI. O JMX ativa o monitoramento e o gerenciamento para os sistemas remotos.

isCatalog
Especifica se o servidor particular hospeda um serviço de catálogo. O valor padrão é `false`.

name
Especifica o nome do servidor.

```
<server
(1) id="server id"
(2) tracespec="the server trace specification"
(3) tracefile="the server trace file"
(4) statspec="the server statistic specification"
(5) jmxport="JMX port number"
(6) isCatalog="true"|"false"
(7) name="the server name"
/>
```

Elemento catalog

Use o elemento catalog para rotear os servidores de contêineres na grade de dados.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

Atributos

host
Especifica o nome do host da estação de trabalho onde o serviço de catálogo está em execução.

port
Especifica o número da porta unido ao nome do host para determinar a porta do serviço de catálogo para a qual o cliente pode se conectar.

```
<catalog
(1) host="catalog service host name"
(2) port="catalog service port number"
/>
```

Elemento container

Utilize o elemento container para armazenar os próprios dados.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

Atributos

objectgridxml
Especifica o caminho e o nome do arquivo do XML descritor a ser usado para especificar as características do ObjectGrid, incluindo mapas, estratégia de bloqueio e plug-ins.

deploymentxml
Especifica o caminho e o nome do arquivo XML que é usado com o descritor XML para determinar o particionamento, a replicação, o número de contêineres iniciais e outras configurações.

server
Especifica o servidor no qual o contêiner está hospedado.

```
<server
(1) objectgridxml="the objectgrid descriptor XML file"
(2) deploymentxml ="the objectgrid deployment descriptor XML file "
(3) server="the server reference "
/>
```

Elemento JPALoader

Use o elemento JPALoader para sincronizar o cache ObjectGrid com um armazenamento de dados backend existente ao usar a API ObjectMap.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

Atributos

entityClassName
Ativa o uso de JPAs, como EntityManager.persist e EntityManager.find. O atributo **entityClassName** é necessário para o JPALoader.

preloadPartition
Especifica a partição na qual o pré-carregamento do mapa será iniciado. Se o valor for menor que 0 ou maior que (totalNumberOfPartition – 1), o pré-carregamento do mapa não será iniciado.

```
<JPALoader
(1) entityClassName="the entity class name"
(2) preloadPartition ="int"
/>
```

Elemento JPATxCallback

Use o elemento JPATxCallback para coordenar as transações JPA e ObjectGrid.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

Atributos

persistenceUnitName
Cria um JPA EntityManagerFactory e localiza os metadados da entidade JPA no arquivo persistence.xml. O atributo **persistenceUnitName** é necessário.

jpaPropertyFactory
Especifica o factory para criar um mapa de propriedade de persistência para substituir as propriedades de persistência padrão. Este atributo deve referenciar um bean.

exceptionMapper
Especifica o plug-in ExceptionMapper que pode ser usado para funções de mapeamento de exceção específicos do JPA ou específicos do banco de dados. Este atributo deve referenciar um bean.

```
<JPATxCallback
(1) persistenceUnitName="the JPA persistence unit name"
(2) jpaPropertyFactory ="JPAPropertyFactory bean reference"
(3) exceptionMapper="ExceptionMapper bean reference"
/>
```

Elemento JPAEntityLoader

Use o elemento JPAEntityLoader para sincronizar o cache ObjectGrid com um armazenamento de dados backend existente ao usar a API EntityManager.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

Atributos

entityClassName
Ativa o uso de JPAs, como EntityManager.persist e EntityManager.find. O atributo **entityClassName** é opcional para o elemento JPAEntityLoader.Se o elemento não estiver configurado, a classe de entidade configurada no mapa de entidade ObjectGrid será usada. A mesma classe deve ser usada para o ObjectGrid EntityManager e para o provedor JPA.

preloadPartition
Especifica o número de partição na qual o pré-carregamento do mapa será iniciado. Se o valor for menor que 0 ou maior que (totalNumberOfPartition – 1), o pré-carregamento do mapa não será iniciado.

```
<JPAEntityLoader
(1) entityClassName="the entity class name"
(2) preloadPartition ="int"
/>
```

Elemento LRUEvictor

Use o elemento LRUEvictor para decidir quais entradas devem ser descartadas quando o mapa exceder o número máximo de entradas.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

Atributos

maxSize
Especifica o total de entradas em uma fila até o evictor tiver que intervir.

sleepTime
Configura a hora em segundos entre o tempo de acesso do evictor sobre as filas de mapa para determinar as ações necessárias no mapa.

numberOfLRUQueues
Permite configurar quantas filas devem ser varridas pelo evictor para evitar que uma única fila tenha o tamanho do mapa inteiro.

useMemoryUsageThresholdEviction
Determina as configurações de liberação com base na quantia de memória usada por uma entrada. O valor padrão é `false`.

```
<LRUEvictor
(1) maxSize="int"
(2) sleepTime ="seconds"
(3) numberOfLRUQueues ="int"
(4) useMemoryUsageThresholdEviction ="true"|"false"
/>
```

Elemento LFUEvictor

Use o elemento LFUEvictor para determinar quais entradas devem se descartadas quando o mapa exceder o número máximo de entradas.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

Atributos

maxSize
Especifica o total de entradas permitidas em cada heap até o evictor tiver que atuar.

sleepTime
Configura a hora em segundos entre o tempo de acesso do evictor sobre os heaps de mapa para determinar as ações necessárias no mapa.

numberOfHeaps
Permite configurar quantos heaps devem ser varridos pelo evictor para evitar que um único heap tenha o tamanho do mapa inteiro.

useMemoryUsageThresholdEviction
Determina as configurações de liberação com base na quantia de memória usada por uma entrada.

```
<LFUEvictor
(1) maxSize="int"
(2) sleepTime ="seconds"
(3) numberOfHeaps ="int"
(4) useMemoryUsageThresholdEviction ="true"|"false"
/>
```

Elemento HashIndex

Use o elemento HashIndex com a reflexão Java para examinar dinamicamente os objetos armazenados em um mapa quando eles forem atualizados.

- Número de ocorrências: Zero para muitas
- Elemento-filho: Nenhum

Atributos

name
Especifica o nome do índice, que deve ser exclusivo para cada mapa.

attributeName
Especifica o nome do atributo a ser indexado. Para índices acessados por campo, o nomes do atributo é equivalente ao nome de campo. Para índices acessados por propriedade, o nomes do atributo é o nome da propriedade compatível com JavaBean.

rangeIndex
Indica se a indexação do intervalo está ativada. O valor padrão é `false`.

fieldAccessAttribute
Usado para mapas sem entidade. O método getter é usado para acessar os dados. O valor padrão é `false`. Se você especificar o valor `true`, o objeto será acessado usando diretamente os campos.

POJOKeyIndex
Usado para mapas sem entidade. O valor padrão é `false`. Se você especificar o valor como `true`, o índice examinará o objeto na parte chave do mapa, que é útil quando a chave for uma chave composta e o valor não tiver a chave integrada com ele. Se você não especificar o valor ou especificá-lo como `false`, então o índice examinará o objeto na parte do valor do mapa.

```
<HashIndex
(1) name="index name"
(2) attributeName="attribute name"
(3) rangeIndex ="true"|"false"
(4) fieldAccessAttribute ="true"|"false"

(5) POJOKeyIndex ="true"|"false"
/>
```

Arquivo `objectgrid.xsd` Spring

Use o arquivo `objectgrid.xsd` Spring para integrar o eXtreme Scale ao Spring para gerenciar as transações do eXtreme Scale e configurar os clientes e servidores.

Consulte o “Arquivo XML descritor do Spring” na página 206 para obter descrições dos elementos e atributos definidos no arquivo `objectgrid.xsd` Spring.

Arquivo `objectgrid.xsd` Spring

```
<xsd:schema xmlns="http://www.ibm.com/schema/objectgrid"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:beans="http://www.springframework.org/schema/beans"
 targetNamespace="http://www.ibm.com/schema/objectgrid"
 elementFormDefault="qualified"
 attributeFormDefault="unqualified">

 <xsd:import namespace="http://www.springframework.org/schema/beans" />

  <xsd:element name="transactionManager">
  <xsd:complexType>
   <xsd:attribute name="id" type="xsd:ID" />
  </xsd:complexType>
 </xsd:element>

  <xsd:element name="register">
  <xsd:complexType>
   <xsd:attribute name="id" type="xsd:ID" />
   <xsd:attribute name="gridname" type="xsd:string" />
  </xsd:complexType>
 </xsd:element>

  <xsd:element name="server">
  <xsd:complexType>
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element ref="catalog" />
   </xsd:choice>
   <xsd:attribute name="id" type="xsd:ID" />
   <xsd:attribute name="tracespec" type="xsd:string" />
   <xsd:attribute name="tracefile" type="xsd:string" />
   <xsd:attribute name="statspec" type="xsd:string" />
   <xsd:attribute name="jmxport" type="xsd:integer" />
   <xsd:attribute name="isCatalog" type="xsd:boolean" />
   <xsd:attribute name="name" type="xsd:string" />
  </xsd: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>
```

Usando o WebSphere Real Time

É possível usar o WebSphere Real Time com o WebSphere eXtreme Scale. Ao ativar o WebSphere Real Time, é possível obter uma coleta de lixo mais previsível com um tempo de resposta e rendimento estável e consistente em um ambiente independente do eXtreme Scale.

Porquê Usar o WebSphere Real Time

O WebSphere eXtreme Scale cria muitos objetos temporários que estão associados a cada transação. Esses objetos temporários lidam com pedidos, respostas, sequências de log e sessões. Sem o WebSphere Real Time, o tempo de resposta de transação pode ultrapassar centenas de milissegundos. Entretanto, usar o WebSphere Real Time com o WebSphere eXtreme Scale pode aumentar a eficiência da coleta de lixo e reduzir o tempo de resposta em 10% do tempo de resposta de configuração independente.

Ativando o WebSphere Real Time

Instale o WebSphere Real Time e o WebSphere eXtreme Scale independente nos computadores onde você planeja executar o eXtreme Scale .Configure a variável de ambiente JAVA_HOME para apontar para o Java SE Runtime Environment (JRE) padrão.

Configure a variável de ambiente JAVA_HOME para apontar para o WebSphere Real Time instalado. Em seguida, ative o WebSphere Real Time da seguinte forma.

1. Edite o arquivo de instalação padrão `objectgridRoot/bin/setupCmdLine.sh | .bat` ao remover o comentário da seguinte linha.

   ```
   WXS_REAL_TIME_JAVA="-Xrealtime -Xgcpolicy:metronome
   -Xgc:targetUtilization=80"
   ```

2. Salve o arquivo.

Agora você ativou o WebSphere Real Time. Se desejar desativar o WebSphere Real Time, poderá incluir o comentário de volta para a mesma linha.

Práticas Recomendáveis

O WebSphere Real Time permite que as transações do eXtreme Scale tenham um tempo de resposta mais previsível. Os resultados mostram que o desvio de um tempo de resposta de uma transação do eXtreme Scale aumenta significativamente com o WebSphere Real Time em comparação com o Java padrão com seu coletor de lixo padrão. A ativação do WebSphere Real Time com o eXtreme Scale é ótima se a estabilidade e o tempo de resposta de seu aplicativo forem essenciais.

As boas práticas descritas nesta seção explicam como tornar o WebSphere eXtreme Scale mais eficiente através do ajuste e de práticas de código dependendo de sua carga esperada.

- Configure o nível de uso correto do processador para o aplicativo e coletor de lixo.

  O WebSphere Real Time fornece a capacidade de controlar o uso do processador para que o impacto da coleta de lixo em seu aplicativo seja controlado e minimizado. Use o parâmetro `-Xgc:targetUtilization=NN` para especificar a porcentagem `NN` do processador que é usado pelo aplicativo a cada 20 segundos. O padrão para WebSphere eXtreme Scale é 80%, mas é possível modificar o script no arquivo `objectgridRoot/bin/setupCmdLine.sh` para configurar um número diferente, como `70`, que fornece mais capacidade de processador para o coletor de lixo. Implemente servidores suficientes para manter a carga do processador abaixo de 80% para seus aplicativos.
- Configure um tamanho maior de memória de heap.

  O WebSphere Real Time usa mais memória do que o Java comum, assim, planeje seu WebSphere eXtreme Scale com uma memória de heap e configure o tamanho de heap ao iniciar os servidores de catálogo e os contêineres com o parâmetro `–jvmArgs –XmxNNNM` no comando ogStartServer. Por exemplo, você pode usar o parâmetro `–jvmArgs –Xmx500M` para iniciar os servidores de catálogo e usar o tamanho de memória apropriado para iniciar os contêineres. O tamanho de memória pode ser configurado de 60 a 70% do tamanho de dados esperado por JVM. Se você não configurar esse valor, ocorrerá um erro OutOfMemoryError. Opcionalmente, use o parâmetro `–jvmArgs –Xgc:noSynchronousGCOnOOM` para evitar um comportamento indesejável quando a JVM ficar sem memória.
- Ajuste os encadeamentos para a coleta de lixo.

  O WebSphere eXtreme Scale cria muitos objetos temporários associados a cada transação e encadeamentos Remote Procedure Call (RPC). A coleta de lixo possui benefícios de desempenho se o computador tiver ciclos de processadores suficientes. O número padrão de encadeamentos é `1`. É possível alterar o número de encadeamentos com o argumento `–Xgcthreads n`. O valor sugerido para esse argumento é o número de núcleos que estão disponíveis em relação ao número de Java virtual machines por computador.
- Ajuste o desempenho para aplicativos de curta duração com oWebSphere eXtreme Scale.

  O WebSphere Real Time é ajustado para aplicativos de execução longa. Geralmente, é necessário executar transações contínuas do WebSphere eXtreme Scale por duas horas para obter dados de desempenho confiáveis. É possível usar o parâmetro `–Xquickstart` para melhorar o desempenho de aplicativos de curta duração. Esse parâmetro informa ao compilador just-in-time (JIT) para usar um nível menor de otimização.
- Minimize a fila do cliente do WebSphere eXtreme Scale e a retransmissão do cliente do WebSphere eXtreme Scale.

A principal vantagem de usar o WebSphere eXtreme Scale com o WebSphere Real Time é que o tempo de resposta da transação é altamente confiável, o que normalmente representa várias vezes melhorias na magnitude de ordem no desvio do tempo de resposta da transação. Todos os pedidos do cliente enfileirados e retransmissões de pedido do cliente através de outro software afetam o tempo de resposta que está além do controle do WebSphere Real Time e do WebSphere eXtreme Scale. É necessário alterar seus encadeamentos e parâmetros de soquete para manter uma carga estável e leve sem nenhum atraso significativo e diminuição da profundidade da fila.

- Escreva aplicativos WebSphere eXtreme Scale para usar o encadeamento do WebSphere Real Time.

  Sem modificar seu aplicativo, é possível obter tempo de reposta de transação WebSphere eXtreme Scale altamente confiável com diversas melhorias da magnitude de ordem no desvio do tempo de resposta. Você pode explorar ainda mais a vantagem do encadeamento de seus aplicativos transacionais a partir do encadeamento Java normal para RealtimeThread o que oferece melhor controle sobre a prioridade de encadeamento e controle de planejamento.

  Seu aplicativo atualmente inclui o código a seguir.

  ```
  public class WXSCacheAppImpl extends Thread implements WXSCacheAppIF
  ```

  Você pode substituir opcionalmente este código pelo seguinte.

  ```
  public class WXSCacheAppImpl extends RealtimeThread implements
  WXSCacheAppIF
  ```

Capítulo 7. Administrando o Ambiente

A administração de um ambiente inclui iniciar e parar servidores em modo independente e dentro do WebSphere Application Server. Também é possível usar o WebSphere eXtreme Scale como um gerenciador de sessões dentro de um ambiente do WebSphere Application Server.

Por Que e Quando Desempenhar Esta Tarefa

Tipos de Servidor

O WebSphere eXtreme Scale possui dois tipos de servidores: *servidores de catálogo* e *servidores de contêiner*. Os servidores de catálogo controlam o posicionamento de shards e descobrem e monitoram os servidores de contêiner. Múltiplos servidores de catálogo juntos compõem o *serviço de catálogo*. Os servidores de contêiner são o Java Virtual Machines que armazenam os dados do aplicativo para a grade.

Modo independente

O modo independente se refere a uma configuração do WebSphere eXtreme Scale que está executando sozinho, sem nenhum outro produto do servidor de aplicativos.

Executando dentro do WebSphere Application Server

Ao executar o WebSphere eXtreme Scale na parte superior do WebSphere Application Server, os servidores de catálogo automaticamente iniciam nos servidores do WebSphere Application Server. Para iniciar os servidores de contêiner, você deve compactar e implementar o seu aplicativo com arquivos XML do ObjectGrid.

Configurando a Disponibilidade de um ObjectGrid

O estado de disponibilidade de uma instância do ObjectGrid determina quais pedidos podem ser processados em qualquer momento específico.

Há quatro estados de disponibilidade:

- ONLINE
- QUIESCE
- OFFLINE
- PRELOAD

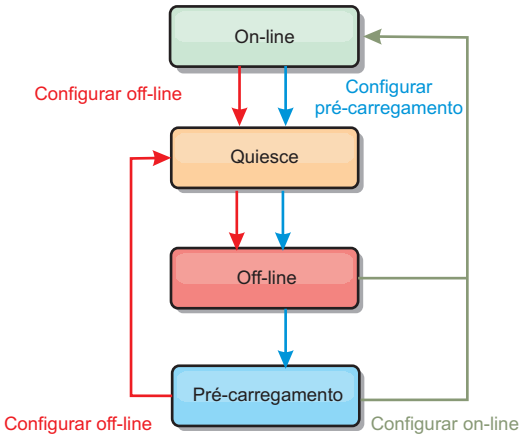


Figura 5. Estados de Disponibilidade de um ObjectGrid

Configurando o Estado de Disponibilidade

O estado padrão de disponibilidade de um ObjectGrid é ONLINE. Um ObjectGrid ONLINE é capaz de processar qualquer pedido de um cliente típico do eXtreme Scale. Entretanto, os pedidos de um cliente de pré-carregamento são rejeitados enquanto o ObjectGrid está ONLINE.

O estado QUIESCE é um estado de transição. Um ObjectGrid que está em QUIESCE, em breve, estará no estado OFFLINE. Enquanto em QUIESCE, um ObjectGrid terá permissão para processar transações pendentes. Entretanto, qualquer nova transação será rejeitada. Um ObjectGrid pode permanecer em QUIESCE por até 30 segundos. Depois deste tempo, o estado de disponibilidade será alterado para OFFLINE.

Um ObjectGrid no estado OFFLINE rejeitará todas as transações.

O estado PRELOAD pode ser utilizado para carregar dados em um ObjectGrid a partir de um cliente de pré-carregamento. Enquanto o ObjectGrid está em um estado PRELOAD, apenas um cliente de pré-carregamento poderá executar o commit em transações junto ao ObjectGrid. Todas as outras transações serão rejeitadas.

Use a interface StateManager para configurar o estado de disponibilidade de um ObjectGrid. Os ObjectGrids do cliente assim como os asObjectGrids do lado do servidor podem ter seu estado de disponibilidade alterado através do uso da interface StateManager. O código a seguir demonstra como alterar o estado de um ObjectGrid cliente.

```
ClientClusterContext client = ogManager.connect("localhost:2809", null, null);
ObjectGrid myObjectGrid = ogManager.getObjectGrid(client, "myObjectGrid");
StateManager stateManager = StateManagerFactory.getStateManager();
stateManager.setObjectGridState(AvailabilityState.OFFLINE, myObjectGrid);
```

Cada shard de ObjectGrid muda para o estado desejado quando o método setObjectGridState é chamado na interface StateManager. Quando o método retornar, todos os shards no ObjectGrid deverão estar no estado adequado.

Use um plug-in ObjectGridEventListener para alterar o estado de disponibilidade de um ObjectGrid do lado do servidor. Altere o estado de disponibilidade de um ObjectGrid do lado do servidor apenas quando o ObjectGrid tiver uma única

partição. Se o ObjectGrid tiver múltiplas partições, o método shardActivated será chamado em cada primário, o que resultará em chamadas desnecessárias para alterar o estado do ObjectGrid

```
public class OGListener implements ObjectGridEventListener, ObjectGridEventGroup.ShardEvents {
    public void shardActivated(ObjectGrid grid) {
            StateManager stateManager = StateManagerFactory.getStateManager();
            stateManager.setObjectGridState(AvailabilityState.PRELOAD, grid);
    }
}
```

Como QUIESCE é o estado transicional, você não pode usar a interface StateManager para colocar um ObjectGrid em um estado QUIESCE. Um ObjectGrid transmite passa por este estado em seu caminho para o estado OFFLINE.

Recuperando o Estado de Disponibilidade

Use o método getObjectGridState da interface StateManager para recuperar o estado de disponibilidade de um ObjectGrid específico.

```
StateManager stateManager = StateManagerFactory.getStateManager();
AvailabilityState state = stateManager.getObjectGridState(inventoryGrid);
```

O método getObjectGridState escolhe um shard primário aleatório no ObjectGrid e retorna seu AvailabilityState. Como todos os shards de um ObjectGrid devem estar no mesmo estado de disponibilidade ou em transição para o mesmo estado de disponibilidade, este método fornece um resultado aceitável para o estado de disponibilidade atual do ObjectGrid.

Estados de Disponibilidade Apropriados para Vários Pedidos

Um pedido será rejeitado se um ObjectGrid não estiver no estado de disponibilidade apropriado para suportar tal pedido. Uma exceção AvailabilityException resulta sempre na rejeição de um pedido.

O atributo initialState

você pode usar o atributo initialState em um ObjectGrid para indicar seu estado de inicialização. Normalmente, quando um ObjectGrid conclui sua inicialização, ele está disponível para roteamento. O estado pode ser posteriormente alterado para evitar o tráfego de roteamento para um ObjectGrid. Se o ObjectGrid precisar ser inicializado, mas não ficar imediatamente disponível, é possível utilizar o atributo initialState.

O atributo initialState é configurado no arquivo XML de configuração do ObjectGrid. O estado padrão é ONLINE. Os valores válidos incluem:

- ONLINE (padrão)
- PRELOAD
- OFFLINE

Conuslte a documentação da API AvailabilityState para obter informações adicionais.

Se initialState estiver configurado em um ObjectGrid, o estado deve ser explicitamente configurado de volta como online ou o ObjectGrid permanecerá indisponível. Ele lançará AvailabilityExceptions.

Utilizando o atributo initialState para pré-carregamento

Se o ObjectGrid for pré-carregado com dados, pode haver um período de tempo entre quando o ObjectGrid está disponível e a mudança para um estado pré-carregado para bloquear o tráfego do cliente. Para evitar este período de tempo, o estado inicial em um ObjectGrid pode ser configurado como PRELOAD. O ObjectGrid ainda conclui todas as inicializações necessárias, mas ele bloqueia o tráfego até que o estado mude e permite que o pré-carregamento ocorra.

Ambos os estados PRELOAD e OFFLINE bloqueiam o tráfego, mas você deve utilizar o estado PRELOAD se deseja iniciar um pré-carregamento.

Comportamento de failover e equilíbrio

Se uma réplica for promovida para um primário, ela não utilizará a configuração initialState. Se o shard primário for movido para um reequilíbrio, a configuração initialState não será usada porque os dados são copiados para o novo local do shard primário antes de concluir a movimentação. Se a replicação não for configurada, então o primário assume o valor initialState se ocorrer failover e um novo primário tiver que ser posicionado.

Iniciando Servidores WebSphere eXtreme Scale Independentes

Ao executar uma configuração do WebSphere eXtreme Scale independente, o ambiente será composto de servidores de catálogo, servidores de contêineres e processos do cliente do eXtreme Scale. É necessário configurar e iniciar estes processos manualmente.

Antes de Iniciar

É possível iniciar os servidores do WebSphere eXtreme Scale em um ambiente que não possua o WebSphere Application Server instalado. Se você estiver utilizando o WebSphere Application Server, consulte “Administrando o WebSphere eXtreme Scale com o WebSphere Application Server” na página 230.

O que Fazer Depois

Pare seus processos do eXtreme Scale. Consulte “Parando Servidores eXtreme Scale Independentes” na página 228 para obter mais informações.

Iniciando o Serviço de Catálogo em um Ambiente Independente

Você deve iniciar o serviço de catálogo manualmente quando estiver usando um ambiente WebSphere eXtreme Scale distribuído que não esteja executando o WebSphere Application Server.

Antes de Iniciar

Se você estiver usando o WebSphere Application Server, o serviço de catálogo inicia automaticamente dentro de um dos processos existentes. Consulte Iniciando o serviço de catálogo no WebSphere Application Server para obter mais informações.

Por Que e Quando Desempenhar Esta Tarefa

O serviço de catálogo pode executar em um processo único ou pode incluir vários servidores de catálogo para formar uma grade do servidor de catálogos. Uma

grade do servidor de catálogo é necessária em um ambiente de produção para alta disponibilidade. Para obter informações adicionais sobre como configurar uma grade do servidor de catálogos, consulte em as informações sobre armazenamento de serviço de catálogo no *Visão Geral do Produto*. O serviço de catálogo, esteja ele posicionado em uma grade ou em um processo único, é iniciado usando o script `startOgServer`. Também é possível especificar parâmetros adicionais para o script vincular o Object Request Broker (ORB) a um host e porta específicos, especificar o domínio ou ativar a segurança.

Ao chamar o comando iniciar, use o script `startOgServer.sh` em plataformas Unix ou `startOgServer.bat` no Windows.

- **Iniciando um único processo de servidor de catálogos**

  Para iniciar um servidor de catálogo único, digite os seguintes comandos a partir da linha de comandos:

  1. Navegue até o diretório bin:
     ```
     cd objectgridRoot/bin
     ```
  2. Execute o comando startOgServer:
     ```
     startOgServer.bat|sh catalogServer
     ```

  Para obter uma lista de todos os parâmetros de linha de comandos disponíveis, consulte "Script startOgServer" na página 223. Não utilize uma única Java Virtual Machine (JVM) para executar o serviço de catálogo em um ambiente de produção. Se o serviço de catálogo falhar, nenhum novo cliente poderá rotear para o eXtreme Scale implementado, e nenhuma nova instância ObjectGrid poderá ser incluída no domínio. Por estas razões, você deve iniciar um conjunto do Java Virtual Machines para executar uma grade do servidor de catálogos.

- **Iniciando uma grade do servidor de catálogos com múltiplos processos**

  Para iniciar um conjunto de servidores para executar um serviço de catálogo, é necessário utilizar a opção **-catalogServiceEndPoints** no script `startOgServer`. Este argumento aceita uma lista de terminais de serviço de catálogo no formato de `serverName:hostName:clientPort:peerPort`. Cada atributo é definido conforme a seguir:

  serverName
  Especifica o nome para identificar o processo que você está ativando.

  hostName
  Especifica o nome do host para o computador onde o servidor é ativado.

  clientPort
  Especifica a porta que é usada para a comunicação da grade do catálogo peer.

  peerPort
  Especifica a porta que é usada para a comunicação da grade do catálogo peer.

  O exemplo a seguir mostra como iniciar a primeira das três Java Virtual Machines para hospedar um serviço de catálogo:

  1. Navegue até o diretório bin:
     ```
     cd objectgridRoot/bin
     ```
  2. Execute o comando startOgServer:
     ```
     startOgServer.bat|sh cs1 -catalogServiceEndpoints
     cs1:MyServer1.company.com:6601:6602,
     cs2:MyServer2.company.com:6601:6602,
     cs3:MyServer3.company.com:6601:6602
     ```

Neste exemplo, o servidor cs1 no host MyServer1.company.com é iniciado. Este nome do servidor é o primeiro argumento que é passado para o script. Durante a inicialização do servidor cs1, os parâmetros catalogServiceEndpoints são examinados para determinar quais portas estão alocadas para este processo. A lista também é utilizada para permitir que o servidor cs1 aceite conexões de outros servidores: cs2 e cs3.

3. Para iniciar os servidores de catálogos restantes na lista, transmita os argumentos a seguir para o script startOgServer. Iniciando o servidor cs2 no host MyServer2.company.com:

```
startOgServer.bat|sh cs2 -catalogServiceEndpoints
cs1:MyServer1.company.com:6601:6602,
cs2:MyServer2.company.com:6601:6602,
cs3:MyServer3.company.com:6601:6602
```

Iniciando cs3 no MyServer3.company.com:

```
startOgServer.bat|sh cs3 -catalogServiceEndpoints
cs1:MyServer1.company.com:6601:6602,
cs2:MyServer2.company.com:6601:6602,
cs3:MyServer3.company.com:6601:6602
```

Importante: Inicie todos os servidores de catálogo em paralelo.

Você deve iniciar os servidores de catálogos que estão em uma grade em paralelo, porque cada servidor faz uma pausa para aguardar até que os outros servidores de catálogo juntem-se ao grupo principal. Um servidor de catálogos que está configurado para uma grade não inicia até identificar outros membros no grupo. O servidor de catálogos eventualmente expira se outros servidores tornam-se disponíveis.

- **Ligando o ORB a um host e porta específicos**

  Não considerando as portas definidas no argumento **catalogServiceEndpoints**, cada serviço de catálogo também utiliza um Object Request Broker (ORB) para aceitar conexões de clientes e contêineres. Por padrão, o ORB atende na porta 2809 do host local. Se desejar vincular o ORB a um host específico em um JVM de serviço de catálogo, use os argumentos **-listenerHost** e **-listenerPort**. O exemplo a seguir mostra como iniciar um único servidor de catálogos JVM com seu ORB ligado à porta 7000 no MyServer1.company.com:

```
startOgServer.sh catalogServer -listenerHost MyServer1.company.com
-listenerPort 7000
```

Cada contêiner e cliente do eXtreme Scale deve ser fornecidos com dados do terminal ORB do serviço de catálogo. Os clientes precisam apenas de um subconjunto destes dados, mas você deve utilizar pelo menos dois terminais para alta disponibilidade.

- **Nomeando o domínio**

  Um nome de domínio não é requerido quando iniciar um serviço de catálogo. O nome de domínio padrão é defaultDomain. Para dar um nome ao seu domínio, use a opção **-domain**. O exemplo a seguir demonstra como iniciar uma única JVM de serviço de catálogo com o nome de domíniomyDomain.

```
startOgServer.sh catalogServer -domain myDomain
```

- **Iniciar um serviço de catálogo seguro**

  É possível iniciar um serviço de catálogo seguro fornecendo os seguintes argumentos:

  -clusterSecurityFile e -clusterSecurityUrl

  Estes argumentos especificam o arquivo objectGridSecurity.xml, que descreve as propriedades de segurança que são comuns a todos os servidores (incluindo servidores de catálogos e servidores de

contêineres). Um dos exemplos de propriedade é a configuração do autenticador que representa o registro do usuário e o mecanismo de autenticação.

-serverProps
Especifica o arquivo de propriedades do servidor, que contém as propriedades de segurança específicas do servidor. O nome do arquivo especificado para esta propriedade está em formato de caminho de arquivo simples, tal como `c:/tmp/og/catalogserver.props`.

Para obter um exemplo de como iniciar um serviço de catálogo seguro, consulte a etapa 2 do do Tutorial de Segurança Java SE no *Visão Geral do Produto*.. Para obter um exemplo do arquivo `objectGridSecurity.xml`, consulte "Arquivo XML Descritor de Segurança" na página 184.

Iniciando Processos do Contêiner

É possível iniciar o eXtreme Scale a partir da linha de comandos usando uma topologia de implementação ou usando um arquivo `server.properties`.

Por Que e Quando Desempenhar Esta Tarefa

Para iniciar um processo de contêiner, é necessário um arquivo XML ObjectGrid. O arquivo XML ObjectGrid especifica quais servidores eXtreme Scale o contêiner hospeda. Assegure-se de que o contêiner esteja equipado para hospedar cada ObjectGrid no XML transmitido para ele. Todas as classes que estes ObjectGrids requerem devem estar no caminho de classe para o contêiner. Para obter mais informações sobre o arquivo XML do ObjectGrid, consulte "Arquivo `objectGrid.xsd`" na página 166.

- **Inicie o processo do contêiner a partir da linha de comandos.**
  1. A partir da linha de comandos, navegue até o diretório bin:
     ```
     cd objectgridRoot/bin
     ```
  2. Execute o seguinte comando:
     ```
     startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

  Importante: No contêiner, a opção **-catalogServiceEndPoints** é utilizada para fazer referência ao host do Object Request Broker (ORB) e porta no serviço de catálogo. O serviço de catálogo utiliza as opções**-listenerHost** e **-listenerPort** para especificar o host e a porta para ligação ORB ou aceita a ligação padrão. Quando estiver iniciando um contêiner, use a opção**-catalogServiceEndPoints** para referenciar os valores que são transmitidos para as opções **-listenerHost** e **-listenerPort** no serviço de catálogo. Se as opções **-listenerHost** e **-listenerPort** não forem utilizadas quando o serviço de catálogo é iniciado, o ORB se conecta à porta 2809 no host local para o serviço de catálogo. Não use a opção **-catalogServiceEndPoints** para referenciar os hosts e portas que foram transmitidos para a opção **-catalogServiceEndPoints** no serviço de catálogo. No serviço de catálogo, a opção **-catalogServiceEndPoints** é usada para especificar portas necessárias para a configuração de servidor estático.
  Esse processo é identificado por c0, o primeiro argumento transmitido para o script. Utilize `companyGrid.xml` para iniciar o contêiner. Se o seu ORB do servidor de catálogos estiver em execução em um host diferente do seu contêiner ou estiver usando uma porta não-padrão, deverá usar o argumento **-catalogServiceEndPoints** para se conectar com o ORB. Para este exemplo, assuma que um único serviço de catálogo está em execução na porta `2809` no `MyServer1.company.com`

- **Inicie o contêiner usando uma política de implementação.**

  Embora não seja necessário, uma política de implementação é recomendada durante a inicialização do contêiner. A política de implementação é utilizada para configurar o particionamento e a replicação para o eXtreme Scale. A política de implementação também pode ser utilizada para influenciar o comportamento de disposição. Como o exemplo anterior não forneceu um arquivo de política de implementação, o exemplo recebe todos os valores-padrão com relação a replicação, particionamento e disposição. Portanto, os mapas no CompanyGrid estão em um mapSet. O mapSet não é particionado ou replicado. Para obter mais informações sobre os arquivos de política de implementação, consulte "Arquivo Descritor XML de Política de Implementação" na página 145. O exemplo a seguir utiliza o arquivo `companyGridDpReplication.xml` para iniciar uma JVM do contêiner, a JVM c0:

  1. A partir da linha de comandos, navegue até o diretório bin:

     ```
     cd objectgridRoot/bin
     ```

  2. Execute o seguinte comando:

     ```
     startOgServer.sh c0 -objectGridFile ../xml/companyGrid.xml
     -deploymentPolicyFile ../xml/companyGridDpReplication.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

  Nota: Se você tiver classes Java armazenadas em um diretório específico, em vez de alterar o script StartOgServer, poderá ativar o servidor com argumentos da seguinte forma: `-jvmArgs -cp C:\ . . . \DirectoryPOJOs\POJOs.jar` . No arquivo `companyGridDpReplication.xml`, um único conjunto de mapas contém todos os mapas. Esse mapSet é dividido em 10 partições. Cada partição possui uma réplica síncrona e nenhuma réplica assíncrona. Qualquer contêiner que utiliza a política de implementação `companyGridDpReplication.xml` em par com o arquivo XML do ObjectGrid `companyGrid.xml` também está apto a hospedar shards do CompanyGrid. Inicie outra JVM do contêiner, a JVM c1:

  1. A partir da linha de comandos, navegue até o diretório bin:

     ```
     cd objectgridRoot/bin
     ```

  2. Execute o seguinte comando:

     ```
     startOgServer.sh c1 -objectGridFile ../xml/companyGrid.xml
     -deploymentPolicyFile ../xml/companyGridDpReplication.xml
     -catalogServiceEndPoints MyServer1.company.com:2809
     ```

  Cada política de implementação contém um ou mais elementos objectgridDeployment. Quando um contêiner é iniciado, ele publica sua política de implementação no serviço de catálogo. O serviço de catálogo examina cada elemento objectgridDeployment. Se o atributo objectgridName corresponde ao atributo objectgridName de um elemento objectgridDeployment anteriormente recebido, o elemento objectgridDeployment mais recente é ignorado. O primeiro elemento objectgridDeployment recebido para um atributo objectgridName especifico é usado como o elemento principal. Por exemplo, assuma que a JVM c2 utiliza uma política de implementação que divide o mapSet em um número diferente de partições:

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>
```

Agora, é possível iniciar uma terceira JVM, a JVM c2:

1. A partir da linha de comandos, navegue até o diretório bin:

   `cd objectgridRoot/bin`

2. Execute o seguinte comando:

```
startOgServer.sh c2 -objectGridFile ../xml/companyGrid.xml
-deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml
-catalogServiceEndPoints MyServer1.company.com:2809
```

O contêiner na JVM c2 é iniciado com uma política de implementação que especifica 5 partições para mapSet1. No entanto, o serviço de catálogo já mantém a cópia principal do objectgridDeployment para o CompanyGrid. Quando a JVM c0 foi iniciada, ela especificou que 10 partições existem para esse mapSet. Como ela foi o primeiro contêiner a iniciar e publicar a política de implementação, sua política de implementação se tornou o principal. Portanto, qualquer valor de atributo objectgridDeployment que seja igual ao `CompanyGrid` em uma política de implementação subsequente será ignorado.

- **Inicie um contêiner utilizando um arquivo de propriedades do servidor.**

  É possível usar um arquivo de propriedades de servidor para configurar o rastreio e configurar a segurança em um contêiner. Execute os seguintes comandos para iniciar o contêiner c3 com um arquivo de propriedades do servidor:

  1. A partir da linha de comandos, navegue até o diretório bin:

     `cd objectgridRoot/bin`

  2. Execute o seguinte comando:

```
startOgServer.sh c3 -objectGridFile ../xml/companyGrid.xml
-deploymentPolicyFile ../xml/companyGridDpReplicationModified.xml
-catalogServiceEndPoints MyServer1.company.com:2809
-serverProps ../serverProps/server.properties
```

A seguir há um exemplo do arquivo `server.properties`:

```
server.properties
workingDirectory=
traceSpec=*=all=disabled
systemStreamToFileEnabled=true
enableMBeans=true
memoryThresholdPercentage=50
```

Este é um arquivo de propriedades de servidor básico que não possui a segurança ativada. Para obter mais informações sobre o arquivo `server.properties`, consulte “Arquivo de Propriedades do Servidor” na página 189.

Script startOgServer

O script startOgServer inicia servidores. É possível utilizar uma variedade de parâmetros quando você inicia seu servidores para ativar o rastreio, especificar números de porta e assim por diante.

Finalidade

É possível utilizar o script startOgServer para iniciar servidores.

Local

O script startOgServer está no diretório bin do diretório-raiz, por exemplo:
`cd objectgridRoot/bin`

Nota: Se você tiver classes Java armazenadas em um diretório específico, em vez de alterar o script StartOgServer, poderá ativar o servidor com argumentos da seguinte forma: `-jvmArgs -cp C:\ . . . \DirectoryPOJOs\POJOs.jar`

.

Uso para Servidores de Catálogos

Para iniciar um servidor de catálogos:

Windows

```
startOgServer.bat <server> [options]
```

UNIX

```
startOgServer.sh <server>[options]
```

Para iniciar um servidor de catálogos configurado padrão, utilize os seguintes comandos:

Windows

```
startOgServer.bat catalogServer
```

UNIX

```
startOgServer.sh catalogServer
```

Opções para Iniciar Servidores de Catálogos

Parâmetros para iniciar um servidor de catálogos:

-catalogServiceEndPoints <server:serverHost:clientPort:peerPort,server:serverHost:clientPort:peerPort>
No contêiner, a opção **-catalogServiceEndpoints** é utilizada para fazer referência ao host do ORB (Object Request Broker) e porta no serviço de catálogo.

-clusterSecurityFile <arquivo xml de segurança do cluster>
Especifica o local do arquivo XML do descritor de segurança.

-clusterSecurityUrl <URL do xml de segurança do cluster>

-domain <nome de domínio>

-listenerHost <nome do host>
Padrão: `localhost`

Especifica o host do listener para comunicação com Internet Inter-ORB Protocol (IIOP).

-listenerPort <porta>
Padrão: `2809`

Especifica a porta listener para comunicação com o IIOP.

-serverProps <arquivo de propriedades do servidor>
Especifica o arquivo de propriedades do servidor que contém as propriedades de segurança específicas do servidor. O nome do arquivo especificado para esta propriedade está simplesmente no formato de caminho de arquivo simples, tal como `c:/tmp/og/catalogserver.props`.

-JMXServicePort <porta>
Padrão: `1099`

Especifica o número da porta para comunicação com Java Management Extensions (JMX).

-traceSpec <especificação de rastreio>
Especifica uma cadeia que especifica o escopo do rastreio que é ativado quando o servidor inicia.

Exemplo:

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <arquivo de rastreio>
Especifica o caminho de um arquivo no qual salvar informações de rastreio.

Exemplo: `../logs/c4Trace.log`

-timeout <segundos>
Especifica um número de segundos antes do servidor expirar.

-script <arquivo de script>

-jvmArgs <argumentos JVM>
Especifica um conjunto de argumentos JVM. Cada parâmetro após o parâmetro **-jvmArgs** é utilizado para iniciar a Java virtual machine (JVM) do servidor. Quando o parâmetro **-jvmArgs** é utilizado, certifique-se de que este seja o último argumento de script opcional especificado.

Exemplo:`-jvmArgs -Xms256M -Xmx1G`

Uso dos Servidores de Contêiner Windows

```
startOgServer.bat <server> -objectgridFile <xml file>
-deploymentPolicyFile <arquivo xml> [options]
```

Windows

```
startOgServer.bat <servidor> -objectgridUrl <URl xml>
-deploymentPolicyUrl <URL xml> [options]
```

UNIX

```
startOgServer.sh <server> -objectgridFile <xml file>
-deploymentPolicyFile <arquivo xml> [options]
```

UNIX

```
 startOgServer.sh <servidor> -objectgridUrl <URL xml>
-deploymentPolicyUrl <URL xml> [options]
```

Opções para Servidores de Contêiner

-catalogServiceEndPoints<hostName:port,hostName:port>
Padrão: localhost:2809

-deploymentPolicyFile <arquivo xml da política de implementação>
Especifica o caminho para o arquivo de políticas de implementação.

Exemplo: `../xml/SimpleDP.xml`

-deploymentPolicyUrl <url do xml da política de implementação>

-listenerHost <nome do host>
Padrão: `localhost`

Especifica o host do listener para comunicação com Internet Inter-ORB Protocol (IIOP).

-listenerPort <porta>
Padrão: `2809`

Especifica a porta listener para comunicação com o IIOP.

-serverProps <arquivo de propriedades do servidor>
Especifica o caminho para o arquivo de propriedades do servidor.

Exemplo:`../security/server.props`

-zone <nome da zona>
Especifica a zona a utilizar para todos os contêineres no servidor.

-traceSpec <especificação de rastreio>
Especifica uma cadeia que especifica o escopo do rastreio que é ativado quando o servidor inicia.

Exemplo:

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <arquivo de rastreio>
Especifica o caminho de um arquivo no qual salvar informações de rastreio.

Exemplo: `../logs/c4Trace.log`

-timeout <segundos>
Especifica um número de segundos antes do servidor expirar.

-script <arquivo de script>

-jvmArgs <argumentos JVM>
Especifica um conjunto de argumentos JVM. Cada parâmetro após o parâmetro **-jvmArgs** é utilizado para iniciar a Java virtual machine (JVM) do servidor. Quando o parâmetro **-jvmArgs** é utilizado, certifique-se de que este seja o último argumento de script opcional especificado.

Exemplo:`-jvmArgs -Xms256M -Xmx1G`

Configurando Portas TCP no Modo Independente

Um Java Virtual Machine que hospeda uma instância de serviço de catálogo requer quatro portas. Duas portas são usadas para uso interno, a terceira porta é usada para clientes e shards de contêiner para se comunicar com o Internet Inter-ORB Protocol (IIOP) e a quarta porta é usada para comunicação com oJava Management Extensions (JMX).

Por Que e Quando Desempenhar Esta Tarefa

Terminais do Serviço de Catálogo do Java Virtual Machine (JVM)

O eXtreme Scale usa o IIOP principalmente para se comunicar com o Java Virtual Machines. O serviço de catálogo do Java Virtual Machines é o único Java Virtual Machines que requer configuração explícita das portas para os serviços IIOP e das portas de serviços de grupo. As portas internas são especificadas usando a opção de linha de comando **-catalogServiceEndPoints**:

```
-catalogServiceEndPoints <server:host:port:port,server:host:port:port>
```

Com a opção de linha de comandos **-catalogServiceEndPoints**, é possível configurar duas portas por servidor. As portas do IIOP são configuradas usando as seguintes opções de linha de comandos:

```
-listenerHost <nome_do_host>
-listenerPort <porta>
```

Quando cada JVM de serviço de catálogo for iniciado, especifique o conjunto completo dos terminais de serviço de catálogo junto com uma porta listener única para essa JVM.

Pontos de extremidade JVM do contêiner

O contêiner do Java Virtual Machines usa duas portas. Uma porta destina-se para uso interno e a outra porta para comunicação IIOP. Normalmente, o contêiner do Java Virtual Machines localiza automaticamente portas não utilizadas e, em seguida, são configuradas para usar duas portas criadas dinamicamente. Esse processo automático minimiza a configuração. Porém, se firewalls estiverem sendo usados ou se você desejar configurar explicitamente as portas, poderá usar uma opção de linha de comandos para especificar a porta do Oject Request Broker (ORB) a ser usada:

```
-listenerHost <nome_do_host>
-listenerPort <porta>
```

Com essas opções de linha de comando, você pode especificar o nome do host (importante para conexão correta com a placa de rede) e a porta a ser especificada para essa JVM. Você pode especificar a porta interna com o argumento da linha de comandos **-haManagerPort**. Porém, para uma configuração mais simples, é possível deixar que o tempo de execução escolha as portas.

1. Inicie o primeiro servidor de catálogos no hostA. A seguir há um exemplo do comando:

   ```
   ./startOgServer.sh cs1 -listenerHost hostA -listenerPort 2809
    -catalogServiceEndPoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
   ```

2. Inicie o segundo servidor de catálogos no hostB. A seguir há um exemplo do comando:

   ```
   ./startOgServer.sh cs2 -listenerHost hostB -listenerPort 2809
    -catalogServiceEndPoints cs1:hostA:6601:6611,cs2:hostB:6601:6611
   ```

   Os clientes precisam apenas saber os terminais do listener do serviço de catálogo. Os clientes recuperam os terminais para o contêiner Java Virtual Machines, que são os Java Virtual Machines que mantêm os dados, automaticamente a partir do serviço de catálogo. Para se conectar com o serviço de catálogo no exemplo anterior, o cliente deve transmitir a seguinte lista de pares `host:port` para a API de conexão:

   ```
   hostA:2809,hostB:2809
   ```

   Para que uma JVM de contêiner use o serviço de catálogo de exemplo, use o seguinte comando:

   ```
   ./startOgServer.sh c0 -catalogServiceEndPoints hostA:2809,hostB:2809
   ```

Parando Servidores eXtreme Scale Independentes

É possível utilizar o script `stopOgServer.sh` para Unix ou o script `stopOgServer.bat` para Windows para parar processos do servidor.

- Parar processos do eXtreme Scale.
  1. A partir da linha de comandos, navegue até o diretório bin.

     ```
     cd objectGridRoot/bin
     ```

  2. Execute o script stopOgServer para parar o servidor. O seguinte exemplo para o servidor c0:

     ```
     stopOgServer.sh c0 -catalogServiceEndPoints MyServer1.company.com:2809
     ```

     Atenção: No contêiner, a opção **-catalogServiceEndPoints** é utilizada para fazer referência ao host do Object Request Broker (ORB) e porta no serviço de catálogo. O serviço de catálogo utiliza as opções **-listenerHost** e **-listenerPort** para especificar o host e a porta para a ligação ORB ou aceita a ligação padrão. Quando estiver parando um contêiner, use a opção **-catalogServiceEndPoints** para referenciar os valores que são transmitidos para as opções **-listenerHost** e **-listenerPort** no serviço de catálogo. Se as opções**-listenerHost** e **-listenerPort** não são utilizadas ao iniciar o serviço de catálogo, o ORB conecta-se à porta 2809 no host local para o serviço de catálogo.

     Não use a opção **-catalogServiceEndPoints** para referenciar os hosts e portas que foram transmitidos para a opção **-catalogServiceEndPoints** no serviço de catálogo. No serviço de catálogo, use a opção **-catalogServiceEndPoints** para especificar portas necessárias para a configuração de servidor estático.
- Parar processos do serviço de catálogo do eXtreme Scale.
  1. A partir da linha de comandos, navegue até o diretório bin.

     ```
     cd objectGridRoot/bin
     ```

  2. Execute o script stopOgServer para parar o servidor.

     ```
     stopOgServer.sh catalogServer -bootstrap MyServer1.company.com:6601
     ```

  Identifique o processo a ser parado com o argumento catalogServer, o primeiro argumento que é transmitido para o script. Para este exemplo, assuma que o serviço de catálogo foi iniciado no domínio MyServer1.company.com com um valor clientAccessPort de 6601.
- Ativar o rastreio para o processo de interrupção do servidor. Se um contêiner falhar ao parar, é possível ativar o rastreio para ajudar com a depuração do problema. Para ativar o rastreio durante a interrupção de um servidor, inclua os parâmetros **-traceSpec** e **-traceFile** nos comandos de interrupção. O parâmetro **-traceSpec** especifica o tipo de rastreio a ser ativado e o parâmetro **-traceFile** especifica o caminho e o nome do arquivo a ser criado e utilizado para os dados de rastreio.
  1. A partir da linha de comandos, navegue até o diretório bin.

     ```
     cd objectGridRoot/bin
     ```

  2. Execute o script stopOgServer com o rastreio ativado.

     ```
     stopOgServer.sh c4 -catalogServiceEndPoints MyServer1.company.com:2809
      -traceFile ../logs/c4Trace.log -traceSpec ObjectGrid=all=enabled
     ```

     Após o rastreio ser obtido, consulte os erros relacionados aos conflitos de porta, às classes ausente, aos arquivos XML ausentes ou incorretos ou a quaisquer rastreios de pilha. As especificações de rastreio de inicialização sugeridas são:

– ObjectGrid=all=enabled
– ObjectGrid*=all=enabled

Para todas as opções de especificação de rastreio, consulte “Opções de Rastreio” na página 281.

Script stopOgServer

O script stopOgServer para servidores.

Finalidade

É possível utilizar o script stopOgServer para parar servidores.

Local

O script stopOgServer está no diretório bin do diretório-raiz, por exemplo:
`cd objectgridRoot/bin`

USO NÃO-RAIZ

Para parar um servidor de catálogos:

Windows

```
stopOgServer.bat <server> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [opções]
```

UNIX

```
stopOgServer.sh <server> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [opções]
```

Para parar um servidor de contêineres do ObjectGrid: Windows

```
stopOgServer.bat <server> -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [opções]
```

UNIX

```
startOgServer.sh -catalogServiceEndPoints
<csHost:csListenerPort,csHost:csListenerPort> [opções]
```

Opções

-clientSecurityFile <arquivo de propriedades de segurança>

-traceSpec <especificação de rastreio>
Especifica uma cadeia que especifica o escopo do rastreio que é ativado quando o servidor inicia.

Exemplo:

- `ObjectGrid=all=enabled`
- `ObjectGrid*=all=enabled`

-traceFile <arquivo de rastreio>
Especifica o caminho de um arquivo no qual salvar informações de rastreio.

Exemplo: `../logs/c4Trace.log`

-jvmArgs <argumentos JVM>
Todos os parâmetros após a opção -jvmArgs são utilizados para iniciar a Java virtual machine (JVM) do servidor. Quando a opção -jvmArgs é utilizada, assegure-se de que seja o último argumento de script opcional especificado.

Administrando o WebSphere eXtreme Scale com o WebSphere Application Server

É possível executar um serviço de catálogo e processos de servidor de contêiner no WebSphere Application Server. O processo para configurar esses servidores é diferente de uma configuração independente. O serviço de catálogo pode ser iniciado automaticamente nos servidores ou gerenciadores de implementação do WebSphere Application Server. O processo do contêiner é iniciado quando um aplicativo eXtreme Scale for implementado e iniciado no ambiente do WebSphere Application Server.

Iniciando o Processo do Serviço de Catálogo em um Ambiente WebSphere Application Server

Um serviço de catálogo único do eXtreme Scale pode iniciar automaticamente em um ambiente WebSphere Application Server ou WebSphere Application Server Network Deployment. É possível configurar o serviço de catálogo para execução em qualquer processo na célula do WebSphere. Um serviço de catálogo único e não armazenado em cluster é aceitável para ambientes de desenvolvimento. Para um ambiente de produção, você deve usar uma grade do serviço de catálogo.

Antes de Iniciar

É necessário instalar o WebSphere Application Server e o WebSphere eXtreme Scale. Consulte Capítulo 4, "Instalando e Implementando o WebSphere eXtreme Scale", na página 27 para obter informações adicionais.

Por Que e Quando Desempenhar Esta Tarefa

O processo do serviço de catálogo é executado nos processos do WebSphere Application Server. Quando o processo que contém o processo do serviço de catálogo é iniciado, o processo de serviço de catálogo também inicia. Para o WebSphere Application Server de base, o serviço de catálogo executa no processo do servidor. Para o WebSphere Application Server Network Deployment, o processo do serviço de catálogo é executado automaticamente no gerenciador de implementação, mas também é possível configurá-lo para executar em um agente do nó ou processo do servidor de aplicativos.

- **Iniciando um único servidor de catálogos no WebSphere Application Server** de base

  Se o WebSphere eXtreme Scale estiver instalado no WebSphere Application Server, o serviço de catálogo inicia automaticamente no processo do servidor. Este recurso simplifica o teste de unidade nos ambientes de desenvolvimento, tais como Rational Application Developer porque não é necessário iniciar explicitamente o serviço de catálogo.

  Se o WebSphere eXtreme Scale estiver instalado no WebSphere Application Server Network Deployment, o serviço de catálogo inicia automaticamente no processo do gerenciador de implementação.

- **Iniciando uma grade do serviço de catálogo no WebSphere Application Server Network Deployment**

  A grade do servidor de catálogos do eXtreme Scale pode ser explicitamente definida em um gerenciador de implementação, agente do nó, ou processo do servidor de aplicativos usando a propriedade customizada catalog.services.cluster, que é definida na célula WebSphere Application Server. Configure o valor da propriedade customizada no seguinte formato:

serverName:*hostName*:*clientPort*:peerPort:*listenerPort*. É possível especificar vários servidores separando os valores com uma vírgula.

Não coloque os serviços de catálogos com contêineres do eXtreme Scale em um ambiente de produção. Inclua o serviço de catálogo em múltiplos processos do agente do nó ou uma grade do servidor de aplicativos que não esteja hospedando um aplicativo eXtreme Scale. Com algumas exceções, a propriedade customizada catalog.services.cluster é similar ao parâmetro **-catalogServiceEndPoints** que é usado ao iniciar um servidor de catálogos independente com a adição do parâmetro **- listenerPort**. Cada atributo para o valor da propriedade customizada é definido conforme a seguir:

serverName
Especifica o nome completo do processo do WebSphere, tal como cellName, nodeName ou serverName, do servidor que hospeda o serviço de catálogo.

Exemplo: `cellA\node1\nodeagent`

hostName
Especifica o nome do servidor de hosting.

clientPort
Especifica a porta que é usada para a comunicação da grade do catálogo peer.

peerPort
Especifica a porta que é usada para a comunicação da grade do catálogo peer.

listenerPort
O listenerPort deve corresponder ao valor BOOTSTRAP_ADDRESS que é definido na configuração do servidor WebSphere.

Para criar a propriedade customizada no console administrativo, clique em **Administração do Sistema** → **Célula** → **Propriedades Customizadas** → **Novo**. Especifique o nome como `catalog.services.cluster` e o valor no formato apropriado, utilizando os atributos definidos. Um exemplo de um valor válido é:

```
cellA\node1\nodeagent:host.local.domain:6600:6601:2809,cellA\node2\
nodeagent:host.foreign.domain:6600:6601:2809
```

Após configurar a propriedade customizada, é necessário reiniciar cada servidor identificado. Ao reiniciar esses servidores, a grade do serviço de catálogo inicia automaticamente.

Restrição: A grade de catálogo usa os mecanismos do grupo principal do WebSphere Application Server. Como resultado, os membros de uma grade de catálogo pode não ampliar os limites de um grupo principal e, portanto, uma grade de catálogo pode não ampliar células. Assim, o eXtreme Scale pode ampliar células conectando-se a um servidor de catálogo por meio dos limites da célula como uma grade de catálogo independente ou como uma grade de catálogo integrada em outra célula.

Iniciando Processos do Contêiner em um Ambiente WebSphere Application Server

Processos de contêiner em um ambiente WebSphere Application Server iniciam automaticamente quando é iniciado um módulo que possui os arquivos XML do eXtreme Scale incluídos.

Antes de Iniciar

WebSphere Application Server e WebSphere eXtreme Scale devem estar instalados e você deve estar apto a acessar o console administrativo do WebSphere Application Server.

Por Que e Quando Desempenhar Esta Tarefa

Aplicativos Java Platform, Enterprise Edition possuem regras de utilitário de classe complexas que complicam muito o carregamento de classes ao utilizar um WebSphere eXtreme Scale compartilhado em um servidor Java EE. Um aplicativo Java EE normalmente é um arquivo Enterprise Archive (EAR) único com um ou mais módulos Enterprise JavaBeans™ (EJB) ou Web archive (WAR) implementados nele.

O WebSphere eXtreme Scale procura cada início de módulo e consulta arquivos XML do eXtreme Scale. Se WebSphere eXtreme Scale detectar que um módulo inicia com os arquivos XML, ele registra o servidor de aplicativos como uma Java Virtual Machine de contêiner do eXtreme Scale (JVM) com o serviço de catálogo. Ao registrar os contêineres com o serviço de catálogo, o mesmo aplicativo pode ser implementado em grades diferentes e ainda ser tratado como uma grade única pelo serviço de catálogo. O serviço de catálogo não é referente às células, grades ou grades dinâmicas. Tudo é uma JVM de contêiner do eXtreme Scale ou não. Uma única grade lógica pode ampliar várias células do WebSphere Extended Deployment, se necessário.

1. Empacote seu arquivo EAR para ter módulos que incluem os arquivos XML do eXtreme Scale na pasta META-INF. O WebSphere eXtreme Scale detecta a presença dos arquivos `objectGrid.xml` e `objectGridDeployment.xml` na pasta META-INF de módulos EJB e da WEB quando eles iniciam. Se apenas um arquivo `objectGrid.xml` for localizado, então a JVM é assumida como sendo o cliente, caso contrário, assume-se que esta JVM atua como um contêiner para a grade que é definida no arquivo `objectGridDeployment.xml`.

   É necessário utilizar os nomes corretos para estes arquivos XML. Os nomes de arquivos fazem distinção entre maiúsculas e minúsculas. Se os arquivos não estiverem presentes, então o contêiner não será iniciado. É possível verificar no arquivo `systemout.log` se há mensagens que indicam que os shards estão sendo localizados. Um módulo EJB ou módulo WAR utilizando o eXtreme Scale deve ter arquivos XML do eXtreme Scale em seu diretório `META-INF`.

   Os arquivos XML do eXtreme Scale incluem:

   - Um arquivo `objectGrid.xml`, comumente referido como um arquivo XML descritor do eXtreme Scale.
   - Um arquivo `objectGridDeployment.xml`, comumente mencionado com um arquivo XML descritor de implementação.
   - Um arquivo `entity.xml`, se forem utilizadas entidades (opcional). O nome do arquivo `entity.xml` deve corresponder ao nome que é especificado no arquivo `objectGrid.xml`.

   O tempo de execução do eXtreme Scale detecta estes arquivos e, em seguida, entra em contato com o serviço de catálogo para informá-lo que outro contêiner está disponível para hospedar shards para tal eXtreme Scale.

   Dica: Se você planeja testar um aplicativo com entidades utilizando um servidor eXtreme Scale, configure o valor minSyncReplicas como `0` no arquivo XML descritor de implementação para evitar uma mensagem `CWPRJ1005E:`

```
Error resolving entity association exception caused by The
EntityMetadata repository is not available. Timeout threshold reached.
```

2. Implementar e iniciar seu aplicativo.

   O contêiner inicia automaticamente quando o módulo é iniciado. O serviço de catálogo inicia para colocar primários e réplicas de partição (shards) o quanto antes. Esta disposição ocorre imediatamente, a menos que você defina um atributo numInitialContainers no arquivo `objectGridDeployment.xml`. Se você definir o atributo numInitialContainers, então a disposição inicia quando tal número de contêineres for iniciado.

O que Fazer Depois

Aplicativos na mesma célula que os contêineres podem conectar-se a estas grades utilizando um método ObjectGridManager.connect(null, null) e, então, chama o método getObjectGrid(ccc, "object grid name"). Os métodos connect ou getObjectGrid podem ser bloqueados até que os contêineres tenham colocado os shards, mas este bloqueio é um problema apenas quando a grade está iniciando.

ClassLoaders

Quaisquer plug-ins ou objetos armazenados em um eXtreme Scale são carregados em um determinado utilitário de carga de classes. Dois módulos EJB no mesmo EAR podem incluir esses objetos. Os objetos são o mesmos, mas são carregados utilizando ClassLoaders diferentes. Se o aplicativo A armazena um objeto Person em um Mapa do eXtreme Scale que é local para o servidor, o aplicativo B recebe um ClassCastException se ele tentar ler tal objeto. Esta exceção ocorre porque o aplicativo B carregou o objeto Person em um utilitário de carga de classe diferente.

Uma abordagem para resolver este problema é ter um módulo raiz que contém os plug-ins e objetos necessários que estão armazenados no eXtreme Scale. Cada módulo que utiliza o eXtreme Scale deve referenciar tal módulo para suas classes. Outra resolução é colocar este objetos compartilhados em um jar utilitário que está em um utilitário de carga de classe comum compartilhado por ambos os módulos e aplicativos. Os objetos também podem ser colocados nas classes do WebSphere ou no diretório `lib/ext`, mas isso dificulta a implementação e não é recomendado.

Os módulos EJB em um arquivo EAR normalmente compartilham o mesmo ClassLoader e não são afetados por este problema. Cada módulo WAR possui seu próprio ClassLoader e é afetado por este problema.

Conectando a uma grade como um cliente apenas

Se a propriedade catalog.services.cluster for definida na célula, nas propriedades customizadas de nó ou servidor, qualquer módulo no arquivo EAR poderá chamar o método ObjectGridManager.connect (ServerFactory.getServerProperties().getCatalogServiceBootstrap(), null, null) para obter um ClientClusterContext e chamar o método ObjectGridManager.getObjectGrid(ccc, "grid name") para obter uma referência para o eXtreme Scale. Se qualquer objeto do aplicativo for armazenado em Mapas, verifique se tais objetos estão presentes em um ClassLoader comum.

Os clientes do Java Platform, Standard Edition ou clientes fora da célula podem conectar-se utilizando a porta IIOP de autoinicialização do serviço de catálogo (o

padrão no Websphere é o gerenciador de implementação) para obter um ClientClusterContext e, em seguida, obter um eXtreme Scale denominado da maneira usual.

Entity Manager

O Entity Manager ajuda porque as Tuplas são armazenadas no Mapas, não os objetos do aplicativo, de forma que há menos problemas com o utilitário de carga de classe. Entretanto, os plug-ins podem ser um problema. Observe também que uma substituição de cliente do arquivo XML descritor do eXtreme Scale sempre é necessária ao conectar-se a um eXtreme Scale que possui entidades definidas: ObjectGridManager.connect(″host:port[,host:port], null, objectGridOverride) ou ObjectGridManager.connect(null, objectGridOverride).

Configurando portas Protocolo de Controle de Transmissões (TCP) em um ambiente WebSphere Application Server

O serviço de catálogo executa uma única instância dentro no gerenciador de implementação, por padrão, e usa a porta de autoinicialização do Internet Inter-ORB Protocol (IIOP) para o gerenciador de implementação do Java Virtual Machine.

Por Que e Quando Desempenhar Esta Tarefa

Os aplicativos da Web ou os aplicativos Enterprise JavaBeans (EJB) em uma célula podem se conectar às grades que estão dentro da mesma célula usando a chamada de conexão `null,null` em vez de especificar as portas de autoinicialização de serviço de catálogo.

Se a grade do serviço de catálogo noWebSphere Application Server for hospedada pelo gerenciador de implementação, os clientes fora da célula (incluindo os clientes Java Platform, Standard Edition) devem se conectar com o serviço de catálogo usando o nome do host do gerenciador de implementação e a porta de autoinicialização IIOP.

Quando o serviço de catálogo for executado nas células do WebSphere Application Server enquanto os clientes forem executados fora das células, será necessário localizar as portas de conexão do cliente ao procurar pelas propriedades customizadas da célula com o nome `catalog.services.cluster`. Se essa entrada for localizada, use-a para se conectar com os clientes ao serviço de catálogo, caso contrário, use a porta IIOP do gerenciador de implementação para a conexão do cliente. Se os clientes estiverem nas células do WebSphere Application Server, será possível recuperar as portas diretamente a partir da interface do CatalogServerProperties.

O eXtreme Scale reutiliza as portas Distribution and Consistency Services (DCS) do gerenciador de alta disponibilidade para associação ao grupo. As portas do Java Management Extensions (JMX) também são reutilizadas.

Gerenciando de Sessões HTTP

Em um ambiente WebSphere Application Server Versão 5 ou posterior, o gerenciador de sessões HTTP que é fornecido com o WebSphere eXtreme Scale pode substituir o gerenciador de sessões padrão no servidor de aplicativos.

O gerenciador de sessões HTTP do WebSphere eXtreme Scale também pode ser executado no WebSphere Application Server Versão 6.0.2 ou posterior em um ambiente que não está executando o WebSphere Application Server, tal comoWebSphere Application Server Community Edition ou Apache Tomcat.

Características

O gerenciador de sessões foi projetado de forma que possa ser executado em qualquer contêiner do Java Platform, Enterprise Edition Versão 1.4. O gerenciador de sessões não possui nenhuma dependência nas APIs do WebSphere, assim é capaz de suportar diversas versões do WebSphere Application Server, bem como ambiente do servidor de aplicativos do fornecedor.

O gerenciador de sessões HTTP fornece recursos de gerenciamento de sessões para um aplicativo associado. O gerenciador de sessões cria sessões HTTP e gerencia os ciclos de vida das sessões HTTP que estão associadas com o aplicativo. Estas atividades de gerenciamento de ciclo de vida incluem: a invalidação de sessões baseada em um tempo limite ou um servlet explícito ou chamada JavaServer Pages (JSP) e a chamada de listeners de sessão que estão associados com a sessão ou ao aplicativo da Web. O gerenciador de sessões persiste suas sessões em uma instância do ObjectGrid. Este instância pode ser uma instância local e de memória ou uma instância totalmente replicada, armazenada em cluster e particionada. O uso da topologia mais recente permite que o gerenciador de sessões forneça suporte de failover da sessão HTTP quando servidores de aplicativos são encerrados ou terminam inesperadamente. O gerenciador de sessões também pode trabalhar em ambientes que não suportam afinidade, quando a afinidade não é forçada por uma camada do balanceador de carga que pulveriza pedidos para a camada do servidor de aplicativos.

Cenários de Uso

O gerenciador de sessões pode ser usado nos seguintes cenários:

- Em ambientes que utilizam servidores de aplicativos em versões diferentes do WebSphere Application Server, tal como em um cenário de migração clássica.
- Em implementações que utilizam servidores de aplicativos de diferentes fornecedores. Por exemplo, um aplicativo que está sendo desenvolvido em servidores de aplicativos de software livre e que estão hospedados no WebSphere Application Server. Outro exemplo é um aplicativo que está sendo promovido do servidor intermediário para produção. A migração contínua destas versões do servidor de aplicativos é possível enquanto todas as sessões HTTP estão ativas e recebendo manutenção.
- Em ambientes que requerem que o usuário persista sessões com níveis de quality of service (QoS) mais altos e melhores garantias de disponibilidade de sessão durante failover de servidor do que os níveis padrão de QoS do WebSphere Application Server.
- Em um ambiente no qual a afinidade de sessão não pode ser garantida, ou ambientes nos quais a afinidade é mantida por um balanceador de carga de fornecedor e o mecanismo de afinidade precisa ser customizado para este balanceador de carga.
- Em um ambiente para transferir o gasto adicional do gerenciamento de sessões e armazenamento para um processo Java externo.
- Em várias células para ativar failover de sessão entre células.

Como o Gerenciador de Sessões Funciona

O gerenciador de sessões se introduz no caminho da solicitação na forma de um filtro de servlet. É possível incluir esse filtro de servlet em cada módulo da Web no seu aplicativo com ferramentas que são enviadas com o WebSphere eXtreme Scale. Também é possível incluir estes filtros manualmente no descritor de implementação da Web do aplicativo. Este filtro recebe o pedido antes do servlet ou dos arquivos JSP no aplicativo de destino. Neste momento, o filtro intercepta os objetos HTTPRequest e HTTPResponse e cria um objeto do wrapper com sua própria implementação.

Esta implementação de filtro intercepta todas as chamadas relacionadas de sessão HTTP que são feitas nos objetos HTTPRequest e HTTPResponse. Estas chamadas são atendidas pelo gerenciador de sessões e não são passados através do gerenciador de sessões base na implementação do servidor Java Platform, Enterprise Edition subjacentes. O gerenciador de sessões cria e mantém sessões e seus ciclos de vida, incluindo tempo limite baseado em invalidações e a ativação de listeners em invalidações de sessão e outros eventos de ciclo de vida.

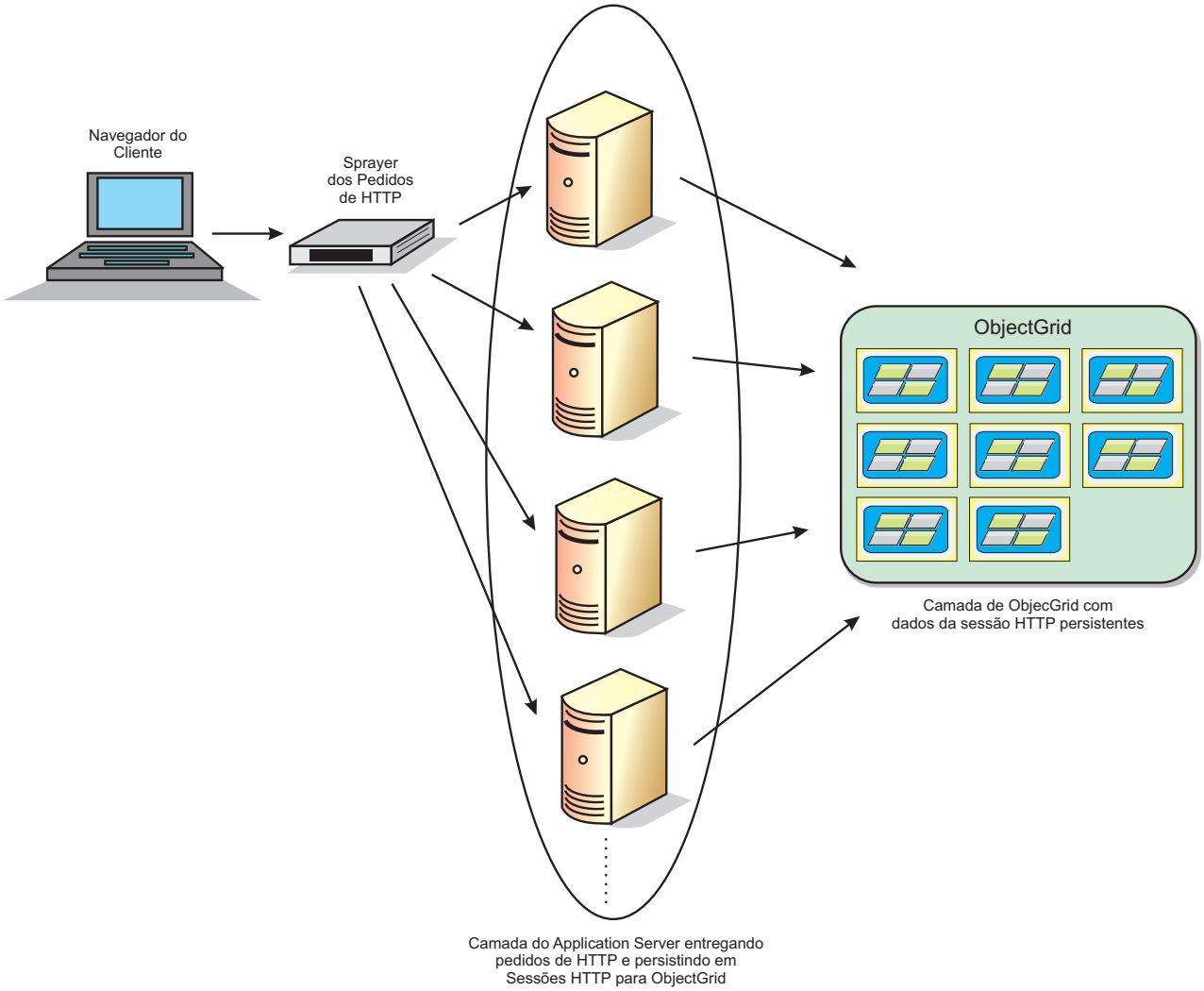


Figura 6. Topologia do Gerenciamento de Sessões HTTP com uma Configuração de Contêiner Remoto

Topologias de Implementação

O gerenciador de sessões pode ser configurado utilizando dois diferentes cenários de implementação dinâmicos:

- **Contêineres do eXtreme Scale conectados por rede integrados**

  Neste cenário, os servidores eXtreme Scale são colocados nos mesmos processos que os servlets. O gerenciador de sessões pode ser comunicar diretamente com a instância do local, evitando atrasos de rede caros.
- **Contêineres do eXtreme Scale conectados por rede remotos**

  Neste cenário, os servidores eXtreme Scale executam em processos externos ao processo no qual os servlets são executados. O gerenciador de sessões comunica-se com um eXtreme Scale remoto.

Recuperando uma Sessão do eXtreme Scale em um Aplicativo Gerenciador de Sessões

```
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);
    }
```

O commit de quaisquer alterações feitas nos mapas com a sessão que são retornadas da chamada do método getAttribute acontece quando ocorre o commit da sessão subjacente.

Configurando o Gerenciador de Sessões do WebSphere eXtreme Scale para Trabalhar com o WebSphere Application Server

Embora o WebSphere Application Server forneça a função de gerenciamento de sessões, este suporte não escala sob carregamentos de pedido extremos. O WebSphere eXtreme Scale é fornecido em um pacote configurável com uma implementação de gerenciamento de sessões que substitui o gerenciador de sessões padrão por um contêiner da Web e fornece melhor escalabilidade e mais opções de configuração robustas.

Por Que e Quando Desempenhar Esta Tarefa

O gerenciador de sessão HTTP WebSphere eXtreme Scale suporta servidores integrados e remotos para armazenamento em cache.

Cenário Integrado

No cenário integrado, os servidores WebSphere eXtreme Scale são colocados no mesmo processo onde os servlets são executados. O gerenciador de sessões pode ser comunicar diretamente com a instância do ObjectGrid local, evitando atrasos de rede caros.

Se você estiver usando o WebSphere Application Server, coloque os arquivos `objectgridRoot/session/samples/objectGrid.xml` e `objectgridRoot/session/samples/objectGridDeployment.xml` fornecidos nos diretórios META-INF do seu arquivo Web archive (WAR). O eXtreme Scale detecta automaticamente esses

arquivos quando o aplicativo é iniciado e inicia automaticamente os contêineres do eXtreme Scale no mesmo processo do gerenciador de sessões.

É possível modificar o arquivo `objectGridDeployment.xml` dependendo se você deseja usar a replicação síncrona ou assíncrona e de quantas réplicas você deseja configurar.

Cenário dos Servidores Remotos

No cenário de servidores remotos, os servidores eXtreme Scale são executados em processos diferentes dos servlets. O gerenciador de sessões comunica-se com um servidor eXtreme Scale remoto. Para utilizar um servidor eXtreme Scale remoto e conectado por rede, o gerenciador de sessões deve ser configurado com os nomes de host e números porta do cluster de serviços de catálogos do eXtreme Scale. O gerenciador de sessões então utiliza uma conexão do cliente do eXtreme Scale para comunicar-se com o servidor de catálogos e os servidores eXtreme Scale.

Se os servidores do eXtreme Scale tiverem que ser iniciados em processos independentes, inicie os contêineres do eXtreme Scale usando os arquivos `objectGridStandAlone.xml` e `objectGridDeploymentStandAlone.xml` que são fornecidos no diretório de amostras do gerenciador de sessões.

1. **Se você estiver executando o WebSphere Application Server ou WebSphere Application Server Network Deployment sem uma instalação do WebSphere eXtreme Scale:** Instale as bibliotecas do eXtreme Scale.

   Copie os arquivos `objectgridRoot/session/lib/sessionobjectgrid.jar` e `objectgridRoot/lib/wsobjectgrid.jar` no caminho de classe para os servidores de aplicativos que hospedam o novo aplicativo ativado pelo gerenciamento de Sessão HTTP. Coloque esses arquivos Java archive (JAR) na pasta`<WAS_HOME>/lib` e em seguida reinicie o servidor para que essas classes estejam visíveis.
2. Una seu aplicativo para que ele possa utilizar o gerenciador de sessões. Para usar o gerenciador de sessões, é necessário incluir as declarações de filtro apropriadas nos descritores de implementação da Web para o aplicativo. Além disso, os parâmetros de configuração do gerenciador de sessões são passados no gerenciador de sessões no formato de parâmetros de inicialização de contexto do servlet nos descritores de implementação. Há três formas nas quais você pode introduzir essas informações no seu aplicativo:
   - **Script addObjectGridFilter**

     Utilize o script de linha de comandos fornecido junto com o eXtreme Scale para unir um aplicativo com declarações de filtro e configuração no formato de parâmetros de inicialização de contexto do servlet. Este script, `objectgridRoot/bin/addObjectGridFilter.sh` ou `objectgridRoot/bin/addObjectGridFilter.bat`, possui dois parâmetros: o aplicativo (caminho absoluto para o arquivo enterprise archive) que precisa ser dividido e o caminho absoluto para o arquivo de propriedades que contém várias propriedades de configuração. O formato de uso deste script é o seguinte:

     Windows

     ```
     addObjectGridFilter.bat [location of ear file] [location of properties file]
     ```

     UNIX

     ```
     addObjectGridFilter.sh [location of ear file] [location of properties file]
     ```

     UNIX **Exemplo utilizando o eXtreme Scale instalado no WebSphere Application Server no Unix:**

a. `cd objectgridRoot/optionalLibraries/ObjectGrid/session/bin`
b. `addObjectGridFilter.sh /tmp/mySessionTest.ear wasRoot/optionalLibraries/ObjectGrid/session/samples/splicer.properties`

UNIX **Exemplo utilizando o eXtreme Scale instalado em um diretório independente no Unix:**

a. `cd objectgridRoot/session/bin`
b. `addObjectGridFilter.sh /tmp/mySessionTest.ear objectgridRoot/session/samples/splicer.properties`

O filtro do servlet que é dividido mantém os padrões para os valores de configuração. É possível substituir estes valores-padrão com opções de configuração especificados no arquivo de propriedades no segundo argumento. Para obter uma lista dos parâmetros que podem ser utilizados, consulte "Parâmetros de inicialização do contexto do servlet" na página 240.

É possível modificar e usar o arquivo `splicer.properties` de amostra que é fornecido com a instalação do eXtreme Scale. Também é possível usar o script addObjectGridServlet, que insere o gerenciador de sessão ao estender cada servlet. Entretanto, o script recomendados é o script addObjectGridFilter.

- **Script de construção Ant**

  O WebSphere eXtreme Scale é fornecido com um arquivo `build.xml` que pode ser utilizado pelo Apache Ant, que está incluído na pasta `wasRoot/bin` de uma instalação do WebSphere Application Server. O `build.xml` pode ser modificado para alterar as propriedades de configuração do gerenciador de sessões. As propriedades de configuração são idênticas aos nomes de propriedades no arquivo `splicer.properties`. Após o `build.xml` ter sido modificado, o processo Ant é chamado ao executar running ant.sh, ws_ant.sh (UNIX) ou ant.bat, ws_ant.bat (Windows).

- **Atualize o descritor da Web manualmente**

  Edite o arquivo `web.xml` que é empacotado com o aplicativo da Web para incorporar a declaração do filtro, seu mapeamento de servlet e os parâmetros de inicialização de contexto do servlet. Você não deve utilizar este método porque ele é propenso a erros.

Para obter uma lista dos parâmetros que podem ser utilizados, consulte "Parâmetros de inicialização do contexto do servlet" na página 240.

3. Implemente o aplicativo. Implemente o aplicativo com seu conjunto normal de etapas para um servidor ou cluster. Após implementar o aplicativo, é possível iniciar o aplicativo.
4. Acesse o aplicativo. Você pode acessar o aplicativo, que interage com o gerenciador de sessões e o WebSphere eXtreme Scale.

O que Fazer Depois

É possível alterar a maioria dos atributos de conexão para o gerenciador de sessões quando você instrumenta seu aplicativo para utilizar o gerenciador de sessões. Esses atributos incluem variações no tipo de replicação (síncrona ou assíncrona), no comprimento de ID de sessão, no tamanho de tabela de sessão de memória, e assim por diante. Não considerando os atributos que podem ser alterados no momento da instrumentação do aplicativo, os únicos outros atributos de configuração que podem ser alterados após a implementação do aplicativo são os atributos que estão relacionados à topologia em cluster do servidor WebSphere eXtreme Scale e a maneira pela qual seus clientes (gerenciadores de sessões) se conectam a eles.

Parâmetros de inicialização do contexto do servlet

A seguinte lista de parâmetros de inicialização do contexto do servlet pode ser especificada no arquivo de propriedades, conforme necessário nos métodos de script ou de divisão baseada em ANT.

Parâmetros

affinityManager
Especifica o nome completo do pacote e da classe Java de um plug-in de filtro do servlet para facilitar a afinidade de roteamento de sessões HTTP. Este valor é ignorado para o WebSphere Application Server porque o suporte à afinidade é integrado. Indique um dos seguintes valores:

- **Sem afinidade (padrão):** com.ibm.ws.httpsession.NoAffinityManager
- **Mecanismo de afinidade do fornecedor:** com.ibm.ws.httpsession.AssumeAffinityManager
- **Construir um novo gerenciador de afinidades:** Utilize a interface com.ibm.wsspi.session.ISessionAffinityManager

persistenceMechanism
Especifica um valor de cadeia que define como a sessão é armazenada no eXtreme Scale. Indique um dos seguintes valores:

- **ObjectGridStore**: Cada atributo session é armazenado como uma entrada diferente na tabela do eXtreme Scale.
- **ObjectGridAtomicSessionStore**: A sessão inteira é armazenada como uma única entrada na tabela do eXtreme Scale.

objectGridName
Especifica um valor de cadeia que define o nome da tabela que é utilizada para um aplicativo da Web específico. O nome padrão vem do nome do aplicativo da Web, conforme derivado do valor ServletContext. A configuração deste parâmetro substitui este valor.

catalogHostPort
Especifica as informações de conexão do servidor de catálogos; o valor precisa estar no formato `host:port<,host:port>`. A lista pode ser arbitrariamente longa e o primeiro endereço viável é utilizado. Esta propriedade é necessária apenas para o cenário do eXtreme Scale remoto e conectado à rede.

replicationType
Um valor de cadeia que define como as atualizações nas sessões são criadas no eXtreme Scale. Os valores válidos são `asynchronous` e `synchronous`. O padrão é `asynchronous`., que é aplicável apenas se a afinidade for utilizada. Caso contrário, apenas synchronous é suportado.

replicationInterval
Um valor de número inteiro que define o tempo em segundos entre a gravação de sessões de atualização no eXtreme Scale quando o valor do parâmetro replicationType é `asynchronous`. O valor-padrão é 10 segundos.

sessionTableSize
Um valor de número inteiro que define o número de sessões que são mantidas na memória com o filtro de servlet além de serem armazenadas no eXtreme Scale. O padrão é 1000.

defaultSessionTimeout
Um valor inteiro que define a quantidade de tempo em minutos que uma sessão pode ficar inativa (por exemplo, não acessada a partir de um servlet) antes da sessão ser invalidada e removida do sistema. O padrão é 30 minutos.

sessionIDLength
Um valor inteiro que define o comprimento dos identificadores de Cadeia que são criados para sessões HTTP. O padrão é 23.

shareSessionsAcrossWebApps
Especifica um valor de cadeia de true ou false. O padrão é false. De acordo com a especificação de servlet, as sessões HTTP não podem ser compartilhadas entre aplicativos da Web. Uma extensão para a especificação do servlet é fornecida para permitir este compartilhamento.

cookieName
Especifica um valor de cadeia que define o nome do cookie para este aplicativo da Web. O padrão é JSESSIONID. Se desejar utilizar um nome de cookie exclusivo, inclua esta propriedade no arquivo `splicer.properties`.

Usando o WebSphere eXtreme Scale para Gerenciamento de Sessão SIP

É possível usar o WebSphere eXtreme Scale como um mecanismo de replicação Session Initiation Protocol (SIP) como uma alternativa confiável ao data replication service (DRS) para replicação de sessão de SIP.

Configuração do Gerenciamento de Sessão de SIP

Para usar o WebSphere eXtreme Scale como o mecanismo de replicação SIP, configure a propriedade customizada com.ibm.sip.ha.replicator.type. No console administrativo, selecione **Servidores de aplicativos** → ***my_application_server*** → **contêiner de SIP** → **Propriedades customizadas** para cada servidor para incluir a propriedade customizada. Digite `com.ibm.sip.ha.replicator.type` para o Nome e `OBJECTGRID` para o Valor.

Utilize as seguintes propriedades para customizar o comportamento do ObjectGrid utilizado para armazenar as sessões de SIP. No console administrativo, clique em **Servidores de aplicativos** → ***my_application_server*** → **Contêiner de SIP** → **Propriedades customizadas** para cada servidor para incluir a propriedade customizada. Digite o **Nome** e **Valor**. Cada servidor deve ter o mesmo conjunto de propriedades para funcionar apropriadamente.

Tabela 8. Propriedades Customizadas para Gerenciamento de Sessão SIP com ObjectGrid

Propriedade	Valor	Padrão
com.ibm.sip.ha.replicator.type	`OBJECTGRID`: use ObjectGrid como armazenamento de sessão SIP	
min.synchronous.replicas	Número mínimo de réplicas síncronas	0
max.synchronous.replicas	Número máximo de réplicas síncronas	0
max.asynchronous.replicas	Número máximo de réplicas assíncronas	1
auto.replace.lost.shards	Consulte “Configuração de Topologia de Implementação” na página 142 para obter informações adicionais.	true
development.mode	• `true` - permite que as réplicas sejam ativadas no mesmo nó das primárias • `false` - as réplicas devem estar em um nó diferente do das primárias	false

Configurando o gerenciador de sessões HTTP para trabalhar com o WebSphere Application Server Community Edition

O WebSphere Application Server Community Edition pode compartilhar estado de sessão, mas não de uma maneira eficiente e escalável. O WebSphere eXtreme Scale fornece uma camada de persistência distribuída e de alto desempenho, que pode ser utilizada para replicar o estado, mas não se integra prontamente a qualquer servidor de aplicativos fora do WebSphere Application Server. É possível integrar estes dois produtos para fornecer uma solução de gerenciamento de sessões escalável. É possível utilizar a infraestrutura modular do WebSphere Application Server Community Edition, GBeans, para incorporar o WebSphere eXtreme Scale como o mecanismo de persistência de estado de sessão.

Antes de Iniciar

É necessário descompactar o arquivo ZIP ou instalar o Geronimo ou o WebSphere Application Server Community Edition e o WebSphere eXtreme Scale em seu sistema.

Por Que e Quando Desempenhar Esta Tarefa

A parte central do WebSphere Application Server Community Edition , o kernel, conta com módulos externos, mencionados como GBeans, para fornecer função e recursos.

1. Distribua os GBeans.
   a. Navegue para o diretório *extremeScale_root*/wasce/bin.
   b. Execute o script addToCeRepository para o sistema que está em execução.
      - Linux UNIX addToRepository.sh ceRoot
      - Windows addToRepository.bat ceRoot
2. Edite o arquivo de configuração. Agora que o GBean foi distribuído, configure o WebSphere Application Server Community Edition para identificar que ele deve ser ativado na inicialização do servidor. É possível registrar o GBean e fornecer a ele argumentos configuráveis que podem ser utilizados no tempo de execução com o arquivo config.xml.
   a. Navegue até o diretório WasCeRoot/var/config.
   b. Abra o arquivo config.xml em um editor de texto simples.
   c. Localize a linha a seguir no arquivo config.xml:

      ```
      <module name="org.apache.geronimo.plugins/mconsole-ds/2.1.1/car"/>
       </attributes>
      ```

   d. Anexe o trecho a seguir antes da tag </attributes>.

      ```
      <module name="com.ibm.websphere.objectgrid/gbean/1.0/car">
         <gbean name="objectgrid/BringupPlugin">
         <attribute name="objectgridHome">c:\objectgrid</attribute>
         <attribute name="geronimoHome">C:\websphereCE</attribute>
         <attribute name="serverName">server1</attribute>
         <attribute name="catalogServiceEndPoints">host:port</attribute>
         <attribute name="replicationDisabled">false</attribute>
         <attribute name="traceSpecification">*=all=disabled</attribute>
         </gbean>
      </module>
      ```

      Os valores para os parâmetros GBean no trecho anterior são exemplos. É possível configurar este valores.

objectgridHome
Especifica o diretório de instalação do objectgridRoot.

geronimoHome -
Especifica o diretório de instalação do WebSphere Application Server Community Edition ou Apache Geronimo.

serverName
Especifica o nome do servidor que deve ser exclusivo para cada WebSphere Application Server Community Edition e a instância do servidor eXtreme Scale

catalogServiceEndPoints -
Especifica o host e a porta do servidor de catálogos eventual.

replicationDisabled -
Especifica se a replicação do estado de sessão está ativada.

traceSpecification -
Especifica uma cadeia para o rastreio do contêiner eXtreme Scale na Java virtual machine (JVM).

3. Inicie um servidor de catálogos. Sua imagem do WebSphere Application Server Community Edition pode estar apta a ativar bem-sucedidamente um servidor eXtreme Scale em sua JVM na inicialização do servidor. Entretanto, para um eXtreme Scale inicializar, primeiro é necessário iniciar um servidor de catálogos. O servidor de catálogos é análogo ao gerenciador de implementação em uma topologia do WebSphere Application Server e deve ser iniciado antes que o WebSphere Application Server Community Edition inicie. Após o servidor de catálogos ser iniciado, insira o nome do host e a porta para o servidor de catálogos no arquivo `config.xml` na entrada do GBean. Consulte "Iniciando o Serviço de Catálogo em um Ambiente Independente" na página 218 para obter mais informações. Certifique-se de ler a seção sobre ligação da porta ORB a um nome do host e porta particulares. É necessário especificar tais terminais na configuração do GBean. Se você tiver um conflito de postas na inicialização do servidor, será necessário abrir o arquivo `ceRoot/var/config/config-substitutions.properties` e alterar a chave NamingPort para um valor diferente de `1099`.
4. Inicie o WebSphere Application Server Community Edition para testar a autoinicialização.
   a. Navegue até o diretório `WasCeRoot/bin`.
   b. Configure a variável de ambiente JAVA_HOME.
      - UNIX Linux `export JAVA_HOME=javaHome`
      - Windows `set JAVA_HOME=javaHome`
   c. Execute o comando start.
      - UNIX Linux `geronimo.sh run`
      - Windows `geronimo.bat run`

   O servidor agora deve inicializar e iniciar o processo de inicialização de seus componentes ou GBeans. O plug-in do eXtreme Scale é o último plug-in a carregar e fornece saídas do rastreio de inicialização necessário e instruções informativas.

Usando o WebSphere eXtreme Scale para a Replicação do Estado de Sessão no WebSphere Application Server Community Edition

Cada aplicativo da Web que é implementado no WebSphere Application Server Community Edition requer conformidade com a especificação do Java Platform, Enterprise Edition, junto com o arquivo descritor `geronimo-web.xml`. Este arquivo contém informações de acesso e dependência específicas no ambiente de tempo de execução para o WebSphere Application Server Community Edition. Para permitir que um aplicativo da Web do Java EE use o recurso de replicação de sessão do WebSphere eXtreme Scale, o aplicativo da Web deverá fornecer dados sobre o plug-in do eXtreme Scale como atributos específicos da sessão.

Antes de Iniciar

Esse processo é simples o bastante para criar em qualquer sistema operacional, sem um IDE ou utilitários.

Por Que e Quando Desempenhar Esta Tarefa

É possível configurar um aplicativo da Web básico que usa a persistência de sessão posterior no eXtreme Scale, que conta e armazena o número de vezes em que um usuário acessou um determinado servlet.

1. Crie o aplicativo da Web de amostra.
   a. Crie um diretório em algum local arbitrário no sistema de arquivos. Por exemplo, é possível criar um diretório chamado *app_home*.
   b. Navegue para o diretório *app_home* e crie a seguinte estrutura de diretórios:

      ```
      > app_home
      --> META-INF
      --> WEB-INF
      -----> lib
      -----> classes
      ```

   c. No diretório *app_home/WEB-INF*, crie um arquivo `web.xml`. Copie e cole o seguinte código no arquivo `web.xml`. Certifique-se de que todos os parâmetros de contexto sejam incluídos no arquivo `web.xml` ou o filtro poderá não funcionar corretamente.

      ```
      <?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. No diretório *app_home/WEB-INF*, crie um arquivo `geronimo-web.xml`. Copie e cole o seguinte código no arquivo `geronimo-web.xml`:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.1"
 xmlns:nam="http://geronimo.apache.org/xml/ns/naming-1.1"
 xmlns:sec="http://geronimo.apache.org/xml/ns/security-1.1"
 xmlns:sys="http://geronimo.apache.org/xml/ns/deployment-1.1">
 <sys:environment>
  <sys:moduleId>
   <sys:groupId>com.ibm.websphere.objectgrid</sys:groupId>
   <sys:artifactId>sample</sys:artifactId>
   <sys:version>1.0</sys:version>
   <sys:type>war</sys:type>
  </sys:moduleId>
  <sys:dependencies>
   <sys:dependency>
    <sys:groupId>com.ibm.websphere.objectgrid
    </sys:groupId>
    <sys:artifactId>session</sys:artifactId>
    <sys:version>1.0</sys:version>
    <sys:type>jar</sys:type>
   </sys:dependency>
  </sys:dependencies>
 </sys:environment>
 <context-root>/sample</context-root>
</web-app>
```

e. Coloque o arquivo `SampleServlet.class` anexado abaixo do diretório *WEB-INF/classes*.

f. Naveque para o diretório *app_home*.

g. Execute o seguinte comando para empacotar o arquivo Web archive (WAR):

```
jar -cf sample.war
```

h. Seu aplicativo da Web agora está pronto para ser implementado.

2. Implemente o aplicativo da Web.

a. Efetue login na página inicial do WebSphere Application Server Community Edition no `hostname:port` especificado. Por padrão, a página inicial é: `http://localhost:8080/`.
b. Clique no **Console Administrativo**.
c. Digite o nome de usuário e a senha e clique em **OK**. Por padrão, as credenciais são:
   - Nome do usuário: System
   - Senha: Manager
d. Selecione **Aplicativos** → **Implementar Novo** no menu no lado esquerdo.
e. Clique em **Procurar** à direita do Archive e selecione o arquivo `sample.war` que você criou na seção anterior.
f. Deixe o campo **Plano** em branco porque você criou o pacote configurável do plano com o aplicativo da Web.
g. Selecione **Iniciar Aplic. após a Instalação**, e clique em **Instalar**.

Depois que o aplicativo for instalado, uma mensagem será exibida informando que a instalação foi bem sucedida.

3. Teste o aplicativo.
   a. Navegue para a raiz de contexto `/sample/SampleServlet` para testar o aplicativo. Por padrão, a raiz de contexto é: `http://localhost:8080/sample/SampleServlet/`.
   b. É necessário visualizar o número de vezes em que você visitou este servlet, seguido pela implementação do HttpSession. É necessário ler o HttpSessionImpl, se ele estiver usando a camada de persistência do eXtreme Scale, ou o StandardSessionFacade, se estiver usando o mecanismo de armazenamento de sessão padrão do contêiner da Web.

Capítulo 8. Monitorando seu Ambiente de Implementação

É possível utilizar APIs, MBeans, logs e utilitários para monitorar o desempenho do seu ambiente de aplicativo.

Visão Geral de Estatísticas

As estatísticas no WebSphere eXtreme Scale são criadas a partir de uma árvore de estatísticas internas. A API do StatsAccessor, os módulos Performance Monitoring Infrastructure (PMI) e a API do MBean são criados a partir da árvore interna.

A seguinte figura mostra a configuração geral das estatísticas para o eXtreme Scale.

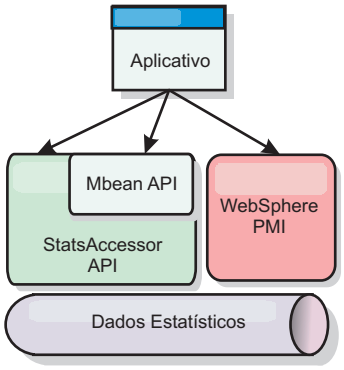


Figura 7. Visão Geral de Estatísticas

Cada uma das APIs oferece uma visualização da árvore de estatísticas, mas são utilizadas por diferentes motivos:

- **API de Estatísticas**: Utilize a API de Estatísticas como um mecanismo de consulta genérico para dados na árvore de estatísticas internas. É possível utilizar a API de Estatísticas para clientes ou aplicativos integrados. Não é possível utilizar a API de Estatísticas se você tiver um eXtreme Scale distribuído.
- **API do MBean**: A API do MBean é um mecanismo baseado em especificação para monitoramento. A API MBean usa a API Statistics e executa localmente para o servidor do JVM (Java Virtual Machine). As estruturas de API e MBean são projetadas para integração imediata com utilitários de outros fornecedores. Utilize a API do MBean quando estiver executando um eXtreme Scale distribuído.
- **WebSphere Application Server Módulos Performance Monitoring Infrastructure (PMI)** : Utilize a PMI se estiver executando o WebSphere eXtreme Scale no WebSphere Application Server. Estes módulos fornecem uma visualização da árvore de estatísticas internas.

API de Estatísticas

Muito semelhante a um mapa de árvore, há um caminho e uma chave correspondentes para recuperar um módulo específico, ou neste caso, nível de granularidade ou agregação. Por exemplo, assume que sempre há um nó-raiz arbitrário na árvore e que as estatísticas estão sendo reunidas para um mapa nomeado como "payroll", pertencendo a um ObjectGrid nomeado como

"accounting". Por exemplo, para acessar o módulo para um nível de agregação ou granularidade do mapa, você poderia passar uma String[] dos caminhos. Neste caso, isto seria igual a String[] {root, "accounting", "payroll"}, já que cada String representaria o caminho do nó. A vantagem desta estrutura é que um usuário pode especificar a matriz para qualquer nó no caminho e obter o nível de agregação para tal nó. Portando, passar String[] {root, "accounting"} forneceria a você as estatísticas do mapa, mas para a grade inteira de "accounting." Isto deixa o usuário tanto com a habilidade de especificar tipos de estatísticas a serem monitorados quanto em que nível de agregação é necessário para o aplicativo.

Módulos PMI do WebSphere Application Server

O WebSphere eXtreme Scale inclui módulos de estatísticas para uso com a PMI do WebSphere Application Server. Quando um perfil do WebSphere Application Server é aumentado com o WebSphere eXtreme Scale, os scripts de aumento integram-se automaticamente aos módulos do WebSphere eXtreme Scale nos arquivos de configuração do WebSphere Application Server. Com a PMI, é possível ativar e desativar módulos de estatísticas, automaticamente agregar estatísticas em várias granularidades e, até mesmo, criar gráficos dos dados utilizando o Tivoli Performance Viewer integrado. Consulte "Monitorando o Desempenho com o PMI do WebSphere Application Server" na página 253 para obter mais informações.

Integração de Produtos de Fornecedores com Beans Gerenciados (MBean)

As APIs do eXtreme Scale e os Beans Gerenciados são projetados para permitir a fácil integração com aplicativos de monitoramento de terceiros. JConsole ou MC4J são alguns exemplos de consoles Java Management Extensions (JMX) leves que podem ser utilizados para analisar informações sobre uma topologia do eXtreme Scale. Também é possível utilizar as APIs programáticas para criar implementações do adaptador para realizar uma captura instantânea ou controlar o desempenho do eXtreme Scale. O WebSphere eXtreme Scale inclui um aplicativo de monitoramento de amostra que permite recursos de monitoramento prontos para utilização e que pode ser utilizado como um modelo para criação de utilitário de monitoramento customizados mais avançados.

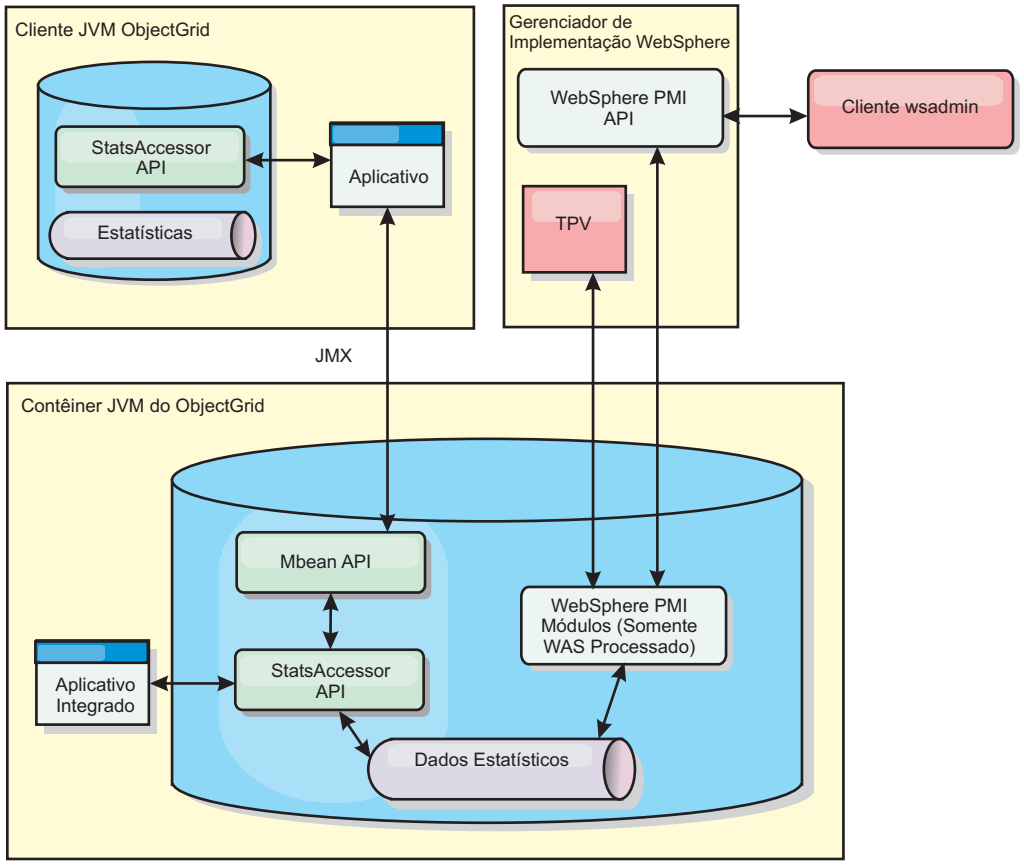


Figura 8. Visão Geral do MBean

Consulte "Utilizando o Utilitário de Amostra xsAdmin" na página 266 para obter mais informações. Para obter informações adicionais sobre a integração com aplicativos de um fornecedor específico, consulte os tópicos a seguir:

- Monitoramento do eXtreme Scale com o IBM Tivoli Monitoring Agent
- "Monitorando o eXtreme Scale com o Hyperic HQ" na página 277
- "Monitorando Aplicativos eXtreme Scale com o CA Wily Introscope" na página 274

Monitorando com a API de Estatísticas

A API de Estatísticas é a interface direta para a árvore de estatísticas interna. As estatísticas são desativadas por padrão, mas podem ser ativadas configurando uma interface StatsSpec. Uma interface StatsSpec define como o WebSphere eXtreme Scale deve monitorar estatísticas.

Por Que e Quando Desempenhar Esta Tarefa

É possível usar a API StatsAccessor local para consultar dados e acessar estatísticas em qualquer instância do ObjectGrid que esteja no mesmo Java Virtual Machine (JVM) que o código de execução. Para obter mais informações sobre as interfaces específicas, consulte a documentação da API. Conclua as seguintes etapas para ativar o monitoramento da árvore de estatísticas internas.

1. Recupere o objeto StatsAccessor. A interface StatsAccessor segue o padrão do singleton. Assim, além dos problemas relacionados ao carregador de classes, uma instância do StatsAccessor deverá existir para cada JVM. Esta classe atua

como a interface principal para todas as operações locais de estatísticas. O seguinte código é um exemplo de como recuperar a classe do acessador. Chame essa operação antes de qualquer outra chamada ObjectGrid.

```
public class LocalClient {

   public static void main(String[] args) {

      // retrieve a handle to the StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

   }

}
```

2. Configure a interface StatsSpec da grade. Configure esse JVM para coletar todas as estatísticas apenas no nível do ObjectGrid. É necessário garantir que um aplicativo ative todas as estatísticas que podem ser necessárias antes de iniciar quaisquer transações. O exemplo a seguir configura a interface StatsSpec utilizando um campo constante estático e utilizando uma spec String. Usar um campo de constante estático é mais simples porque o campo já definiu a especificação. Entretanto, ao utilizar uma spec String, é possível ativar qualquer combinação de estatísticas que são necessárias.

```
public static void main(String[] args) {

      // retrieve a handle to the StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Set the spec via the static field
      StatsSpec spec = new StatsSpec(StatsSpec.OG_ALL);
      accessor.setStatsSpec(spec);

      // Configurar o spec através do spec String
      StatsSpec spec = new StatsSpec("og.all=enabled");
      accessor.setStatsSpec(spec);

}
```

3. Envie as transações para a grade para forçar a coleta dos dados para monitoramento. Para coletar dados úteis para as estatísticas, é necessário enviar as transações para a grade. O seguinte extrato de código insere um registro no MapA, que é um ObjectGridA. Como as estatísticas estão no nível do ObjectGrid, qualquer mapa dentro do ObjectGrid produz os mesmos resultados.

```
public static void main(String[] args) {

      // retrieve a handle to the StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Set the spec via the static field
      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");

      // Drive insert
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();
}
```

4. Consulte um StatsFact utilizando a API do StatsAccessor. Cada caminho de estatísticas é associado com uma interface StatsFact. A interface StatsFact é um marcador genérico que é utilizado para organizar e conter um objeto StatsModule. Antes de poder acessar o módulo de estatísticas real, o objeto StatsFact deve ser recuperado.

```
public static void main(String[] args) {

      // retrieve a handle to the StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Set the spec via the static field
      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");

      // Drive insert
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();

      // Retrieve StatsFact

      StatsFact fact = accessor.getStatsFact(new String[] {"EmployeeGrid"},
    StatsModule.MODULE_TYPE_OBJECT_GRID);

}
```

5. Interaja com o objeto StatsModule. O objeto StatsModule está contido na interface StatsFact. É possível obter uma referência para o módulo utilizando a interface StatsFact. Como a interface StatsFact é uma interface genérica, é necessário converter o módulo retornado para o tipo StatsModule esperado. Como esta tarefa coleta estatísticas do eXtreme Scale, o objeto StatsModule retornado é convertido para um tipo OGStatsModule. Após o módulo ser convertido, é possível acessar todas as estatísticas disponíveis.

```
public static void main(String[] args) {

      // retrieve a handle to the StatsAccessor
      StatsAccessor accessor = StatsAccessorFactory.getStatsAccessor();

      // Set the spec via the static field
      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");

      // Drive insert
      session.begin();
      map.insert("SomeKey", "SomeValue");
      session.commit();

      // Retrieve StatsFact
      StatsFact fact = accessor.getStatsFact(new String[] {"EmployeeGrid"},
    StatsModule.MODULE_TYPE_OBJECT_GRID);

      // Retrieve module and time
      OGStatsModule module = (OGStatsModule)fact.getStatsModule();
      ActiveTimeStatistic timeStat =
    module.getTransactionTime("Default", true);
      double time = timeStat.getMeanTime();

}
```

Módulos Estatísticos

O WebSphere eXtreme Scale usa um modelo de estatísticas interno para controlar e filtrar dados, que é a estrutura subjacente usada por todas as visualizações de dados para reunir capturas instantâneas das estatísticas. É possível utilizar diversos métodos para recuperar as informações dos módulos estatísticos.

Visão Geral

As estatísticas no WebSphere eXtreme Scale são controladas e contidas nos componentes StatsModules. No modelo estatístico, existem diversos tipos de módulos estatísticos:

OGStatsModule
Fornece estatísticas para uma instância do ObjectGrid, incluindo tempos de resposta da transação.

MapStatsModule
Fornece estatísticas para um único mapa, incluindo o número de entradas e a taxa de ocorrências.

QueryStatsModule
Fornece estatísticas para consultas, incluindo criação de planos e tempo de execução.

AgentStatsModule
Fornece estatísticas para agentes de API do DataGrid API, incluindo tempos de de serialização e tempos de execução.

HashIndexStatsModule
Fornece estatísticas para consulta HashIndex e tempos de execução de manutenção.

SessionStatsModule
Fornece estatísticas ao plug-in do gerenciador de sessões HTTP.

Para obter detalhes dobre os módulos estatísticos, consulte o pacote com.ibm.websphere.objectgrid.stats na Documentação da API .

Estatísticas em um Ambiente Local

O modelo é organizado como uma árvore n-ária (uma estrutura em árvore com o mesmo grau para todos os nós) que é composta de todos os tipos de StatsModule mencionados na lista anterior. Devido a esta estrutura de organização, cada nó na árvore é representado pela interface StatsFact. A interface StatsFact pode representar um módulo individual ou um grupo de módulos para propósitos de agregação. Por exemplo, se vários nós-folhas na árvore representarem objetos MapStatsModule específicos, o nó-pai StatsFact para estes nós conterá estatísticas agregadas para todos os módulos-filhos. Após buscar um objeto StatsFact, é possível utilizar a interface para recuperar o StatsModule correspondente.

Muito semelhante a um mapa de árvore, é utilizado um caminho ou chave correspondente para recuperar um StatsFact específico. O caminho é um valor String[] que consiste em cada nó que está ao long do caminho para o fato solicitado. Por exemplo, você criou um ObjectGrid denominado ObjectGridA, que contém dois Mapas: MapA e MapB. O caminho para o StatsModule para MapA seria semelhante a `[ObjectGridA, MapA]`. O caminho para as estatísticas agregadas para ambos os mapas seria: `[ObjectGridA]`.

Estatísticas em um Ambiente Distribuído

Em um ambiente distribuído, os módulos de estatísticas são recuperados utilizando um caminho diferente. Como um servidor pode conter várias partições, a árvore de estatísticas precisa controlar a partição a qual cada módulo pertence. Como resultado, o caminho para consultar um objeto StatsFact específico é diferente. Utilizando o exemplo anterior, mas incluindo nele os mapas existentes na partição 1, o caminho é `[1, ObjectGridA, MapA]` para recuperação de tal objeto StatsFact para MapA.

Monitorando o Desempenho com o PMI do WebSphere Application Server

O WebSphere eXtreme Scale suporta Performance Monitoring Infrastructure (PMI) ao executar em um WebSphere Application Server ou servidor de aplicativos WebSphere Extended Deployment. A PMI coleta dados de desempenho em aplicativos de tempo de execução e fornece interfaces que suportam aplicativos externos para monitorar dados de desempenho. É possível utilizar o console administrativo ou a ferramenta wsadmin para acessar dados de monitoramento.

Antes de Iniciar

- É possível utiliza a PMI para monitorar seu ambiente quando você está utilizando oWebSphere eXtreme Scale combinado com o WebSphere Application Server.

Por Que e Quando Desempenhar Esta Tarefa

O WebSphere eXtreme Scale utiliza o recurso PMI customizado do WebSphere Application Server para incluir sua própria instrumentação PMI. Com esta abordagem, é possível ativar e desativar a PMI do WebSphere eXtreme Scale com o console administrativo ou com as interfaces Java Management Extensions (JMX) na ferramenta wsadmin. Além disso, é possível acessar as estatísticas do WebSphere eXtreme Scale com a PMI padrão e as interfaces JMX que são utilizadas pela ferramentas de monitoramento, incluindo o Tivoli Performance Viewer.

1. Ative a PMI do eXtreme Scale. É necessário ativar a PMI para visualizar as estatísticas de PMI. Consulte "Ativando a PMI" para obter mais informações.
2. Recupere as estatísticas de PMI do eXtreme Scale. Visualize o desempenho dos seus aplicativos eXtreme Scale com o Tivoli Performance Viewer. Consulte "Recuperando Estatísticas PMI" na página 255 para obter mais informações.

O que Fazer Depois

Para obter mais informações sobre a ferramenta wsadmin, consulte "Utilizando o Utilitário wsadmin" na página 264.

Ativando a PMI

É possível utilizar a WebSphere Application Server Performance Monitoring Infrastructure (PMI) para ativar ou desativar estatísticas em qualquer nível. Por exemplo, você pode optar por ativar as estatísticas de taxa de acesso do mapa para um mapa específico, mas não o número de estatísticas de entrada ou as estatísticas de tempo de atualização de lote do utilitário de carga. É possível ativar a PMI no console administrativo ou com scripts.

Antes de Iniciar

Seu servidor de aplicativos deve ser iniciado e ter um aplicativo ativado para o eXtreme Scale instalado. Para ativar a PMI com scripts, você também deve estar apto a efetuar login e utilizar a ferramenta wsadmin. Para obter informações adicionais sobre a ferramenta wsadmin, consulte o tópico ferramenta wsadmin no centro de informações do WebSphere Application Server.

Por Que e Quando Desempenhar Esta Tarefa

Utilize a PMI do WebSphere Application Server para fornecer um mecanismo granular com o qual é possível ativar ou desativar estatísticas em qualquer nível. Por exemplo, você pode optar por ativar as estatísticas de taxa de acesso do mapa para um mapa específico, mas não o número de estatísticas de entrada ou as estatísticas de tempo de atualização de lote do utilitário de carga. Esta seção mostra como utilizar o console administrativo e os scripts wsadmin para ativar a PMI do ObjectGrid.

- **Ativar a PMI no console administrativo.**
  1. No console administrativo, clique em **Monitoramento e Ajuste** → **Performance Monitoring Infrastructure** → *server_name*.
  2. Verifique se opção Ativar PMI (Performance Monitoring Infrastructure) está selecionada. Essa definição é ativada por padrão. Se a configuração não estiver ativada, selecione a caixa de opções e reinicie o servidor.
  3. Clique em **Customizar**. Na árvore de configuração, selecione o ObjectGrid e o módulo Mapas do ObjectGrid. Ative as estatísticas para cada módulo.

  A categoria de tipo de transação para estatísticas do ObjectGrid é criada no tempo de execução. É possível visualizar apenas as subcategorias das estatísticas do ObjectGrid e do Mapa na guia **Tempo de Execução**.
- **Ativar a PMI com scripts.**
  1. Abra um prompt de linha de comandos. Navegue para o diretório install_root/bin. Digite wsadmin para iniciar a ferramenta de linha de comandos wsadmin.
  2. Modifique a configuração do tempo de execução do PMI do eXtreme Scale. Verifique se a PMI está ativada para o servidor, por meio dos seguintes comandos:

```
wsadmin>set s1 [$AdminConfig getid /Cell:CELL_NAME/Node:NODE_NAME/Server: APPLICATION_SERVER_NAME/]
wsadmin>set pmi [$AdminConfig list PMIService $s1]
wsadmin>$AdminConfig show $pmi.
```

     Se a PMI não estiver ativada, execute os seguintes comandos para ativá-la:

```
wsadmin>$AdminConfig modify $pmi {{enable true}}
wsadmin> $AdminConfig save
```

     Se precisar ativar a PMI, reinicie o servidor.
  3. Configure as variáveis para customizar o conjunto de estatísticas utilizando os seguintes comandos:

```
wsadmin>set perfName [$AdminControl completeObjectName type=Perf,
process=APPLICATION_SERVER_NAME,*]
wsadmin>set perfOName [$AdminControl makeObjectName $perfName]
wsadmin>set params [java::new {java.lang.Object[]} 1]
wsadmin>$params set 0 [java::new java.lang.String custom]
wsadmin>set sigs [java::new {java.lang.String[]} 1]
wsadmin>$sigs set 0 java.lang.String
```

  4. Configure o conjunto de estatísticas para customizar o seguinte comando:

```
wsadmin>$AdminControl invoke_jmx $perfOName setStatisticSet $params $sigs
```

  5. Utilize os comandos a seguir para configurar as variáveis de modo a ativar a estatística da PMI objectGridModule:

```
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. Configure a cadeia de estatísticas por meio deste comando:

```
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. Configure a cadeia de estatísticas por meio deste comando:

```
wsadmin>$AdminControl invoke_jmx $perfOName setCustomSetString $params2 $sigs2
```

Essas etapas ativam o PMI do tempo de execução do eXtreme Scale, mas não modificam a configuração do PMI. Se você reiniciar o servidor de aplicativos, as configurações da PMI serão perdidas, exceto para a ativação da PMI principal.

Exemplo

Você pode executar as seguintes etapas para ativar as estatísticas da PMI para o aplicativo de amostra:

1. Ative o aplicativo utilizando o endereço da Web `http://host:port/ObjectGridSample`, em que host e porta são o nome do host e número de porta HTTP do servidor no qual a amostra está instalada.
2. No aplicativo de amostra, clique consecutivamente em ObjectGridCreationServlet e nos botões de ação 1, 2, 3, 4 e 5 para gerar ações para o ObjectGrid e os mapas. Não feche esta página do servlet neste momento.
3. No console administrativo, clique em **Monitoramento e Ajuste** → **Performance Monitoring Infrastructure** → *server_name* Clique na guia **Tempo de Execução**.
4. Clique no botão de rádio **Customizado**.
5. Expanda o módulo Mapas do ObjectGrid na árvore de tempo de execução e clique no link clusterObjectGrid. No grupo Mapas do ObjectGrid, há uma instância do ObjectGrid denominada clusterObjectGrid; há quatro mapas debaixo do grupo clusterObjectGrid: contadores, funcionários, escritórios e sites. Na instância do ObjectGrids, existe uma instância do clusterObjectGrid e sob esta há um tipo de transação denominado DEFAULT.
6. Você pode ativar as estatísticas de seu interesse. Por exemplo, é possível ativar o número de entradas para o mapa funcionários e o tipo de resposta de transação para o tipo de transação DEFAULT.

O que Fazer Depois

Quando ativar a PMI, será possível visualizar estatísticas de PMI com o console administrativo ou por meio de scripts.

Recuperando Estatísticas PMI

Ao recuperar estatísticas PMI, é possível visualizar o desempenho dos aplicativos eXtreme Scale.

Antes de Iniciar

- Ative o controle de estatísticas PMI para o seu ambiente. Consulte "Ativando a PMI" na página 253 para obter mais informações.

- Os caminhos nesta tarefa assumem que você está recuperando estatísticas para o aplicativo de amostra, mas é possível utilizar estas estatísticas para qualquer outro aplicativo com etapas semelhantes.
- Se estiver utilizando o console administrativo para recuperar estatísticas, você deve estar apto a efetuar login no console administrativo. Se estiver utilizando scripts, você deve estar apto a efetuar login no wsadmin.

Por Que e Quando Desempenhar Esta Tarefa

É possível recuperar estatísticas PMI para visualização no Tivoli Performance Viewer concluindo etapas no console administrativo ou com scripts.

- Etapas do console administrativo
- Etapas de scripts

Para obter mais informações sobre as estatísticas que podem ser recuperadas, consulte "Módulos PMI" na página 257.

- Recupere estatísticas PMI no console administrativo.
  1. No console administrativo, clique em **Monitoramento e Ajuste** → **Performance Viewer** → **Atividade Atual**
  2. Selecione o servidor que deseja monitorar utilizando o Tivoli Performance Viewer, em seguida, ative o monitoramento.
  3. Clique no servidor para visualizar a página do Performance Viewer.
  4. Expanda a árvore de configuração. Clique em **Mapas do ObjectGrid** → **clusterObjectGrid** e selecione **employees**. Expanda **ObjectGrids** → **clusterObjectGrid** e selecione **DEFAULT**.
  5. No aplicativo de amostra do ObjectGrid, retorne ao servlet ObjectGridCreationServlet, clique no botão 1, em seguida, ocupar mapas. Você pode visualizar a estatística no visualizador.
- Recupere estatísticas PMI com scripts.
  1. Em um prompt de linha de comandos, navegue até o diretório `install_root/bin`. Digite `wsadmin` para iniciar a ferramenta wsadmin.
  2. Configure as variáveis para o ambiente utilizando os seguintes comandos:

     ```
     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. Configure as variáveis para obter o ambiente mapModule utilizando os seguintes comandos:

     ```
     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. Obtenha a estatística mapModule utilizando os seguintes comandos:

     ```
     wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params $sigs
     ```

  5. Configure as variáveis para obter a estatística objectGridModule utilizando os seguintes comandos:

     ```
     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. Obtenha a estatística objectGridModule utilizando os seguintes comandos:

```
wsadmin>$AdminControl invoke_jmx $perfOName getStatsString $params2 $sigs2
```

Resultados

É possível visualizar estatísticas no Tivoli Performance Viewer.

Módulos PMI

É possível monitorar o desempenho dos seus aplicativos com os módulos Performance Monitoring Infrastructure (PMI).

objectGridModule

O objectGridModule contém uma estatística de tempo: tempo de resposta de transação. Uma transação é definida como uma duração entre a chamada de método Session.begin e a chamada de método Session.commit. Esta duração é rastreada como o tempo de resposta da transação. O elemento-raiz do objectGridModule, "root", serve como ponto de entrada para as estatísticas do WebSphere eXtreme Scale. Este elemento-raiz possui ObjectGrids como seus elementos-filhos, que possuem tipos de transações como seus elementos-filhos. A estatística de tempo de resposta está associada a cada tipo de transação.

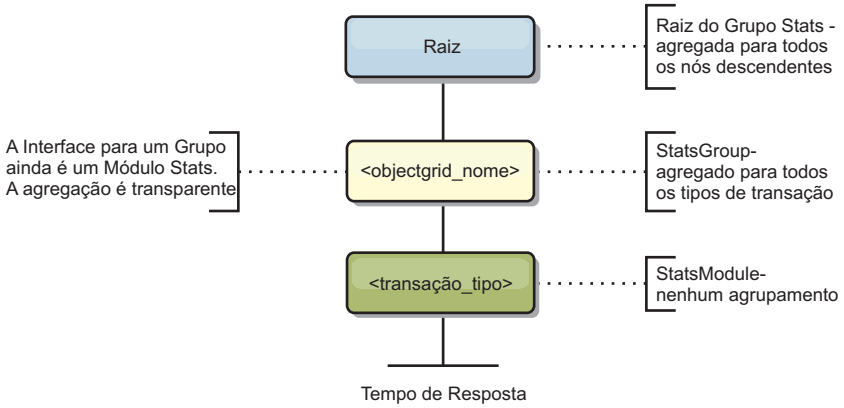


Figura 9. Estrutura do Módulo ObjectGridModule

O seguinte diagrama mostra um exemplo da estruturaObjectGridModule. Neste exemplo, duas instâncias ObjectGrid existem no sistema: ObjectGrid A e ObjectGrid B. A instância ObjectGrid A possui dois tipos de transações: A e padrão. A instância ObjectGrid B possui apenas o tipo de transação padrão.

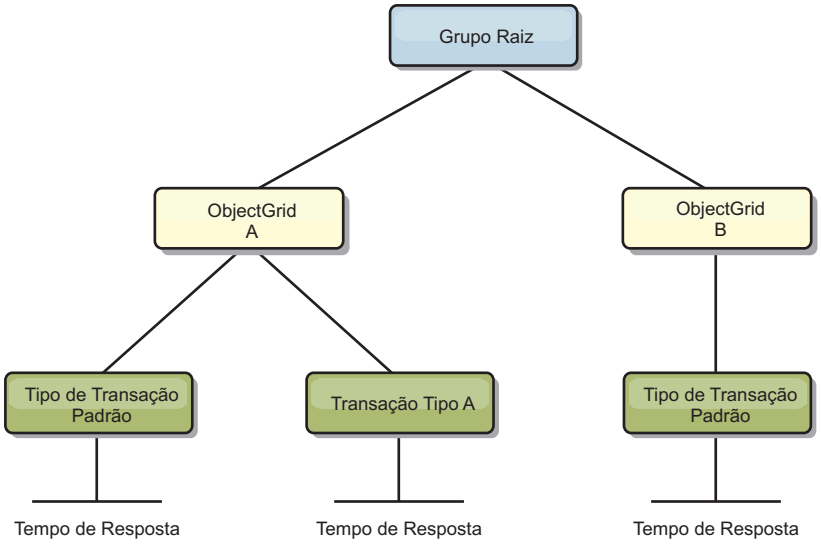


Figura 10. Exemplo da Estrutura do Módulo ObjectGridModule

Os tipos de transações são definidos por desenvolvedores de aplicativos, porque eles sabem quais tipos de transações seus aplicativos utilizam. O tipo de transação é configurando utilizando o método Session.setTransactionType(String) a seguir:

```
/**
* Configura o tipo de transação para futuras transações.
*
* Quando este método é chamado, todas as futuras transações terão o mesmo tipo
* até que outro tipo de transação seja configurado. Se nenhum tipo de transação
* estiver configurado, será utilizado o tipo de transação TRANSACTION_TYPE_DEFAULT
* padrão.
*
* Os tipos de transações são utilizados principalmente para finalidade de rastreamento de dados estatísticos.
* Os usuários podem predefinir tipos de transações que são executadas em um
* application. A idéia é categorizar transações com as mesmas características
* para uma categoria (tipo), portanto, uma estatística de tempo de resposta de transação pode ser
* utilizada para rastrear cada tipo de transação.
*
* Este rastreamento é útil quando seu aplicativo tem diferentes tipos de
* transações.
* Entre eles, alguns tipos de transações, como transações de atualização, têm um processamento
* mais longo que outras transações, como transações de leitura. Utilizando o tipo de
* transação, diferentes transações são rastreadas por diferentes estatísticas, portanto,
* as estatísticas podem ser mais úteis.
*
* @param tranType o tipo de transação para futuras transações.
*/
void setTransactionType(String tranType);
```

O exemplo a seguir configura o tipo de transação como `updatePrice`:

```
// Set the transaction type to updatePrice
// The time between session.begin() and session.commit() will be
// tracked in the time statistic for "updatePrice".
session.setTransactionType("updatePrice");
session.begin();
map.update(stockId, new Integer(100));
session.commit();
```

A primeira linha indica que o tipo de transação subseqüente é updatePrice. Uma estatística updatePrice existe na instância ObjectGrid que corresponde à sessão no exemplo. Utilizando interfaces Java Management Extensions (JMX), você pode obter o tempo de resposta da transação para transações updatePrice. Também é possível obter a estatística agregada para todos os tipos de transações na instância ObjectGrid especificada.

mapModule

O mapModule contém três estatísticas relacionadas aos mapas eXtreme Scale:

- **Taxa de ocorrências do mapa** - *BoundedRangeStatistic*: Controla a taxa de ocorrências de um mapa. A taxa de ocorrência é um valor flutuante entre 0 e 100 inclusivamente, que representa a porcentagem de ocorrências do mapa em relação a operações get do mapa.
- **Número de entradas**-*CountStatistic*: Controla o número de entradas no mapa.
- **Tempo de resposta de atualização de lote do utilitário de carga**-*TimeStatistic*: Controla o tempo de resposta que é utilizada para a operação de atualização de lote do utilitário de carga.

O elemento-raiz do mapModule, "root", serve como ponto de entrada para as estatísticas do Mapa ObjectGrid. Este elemento-raiz possui ObjectGrids como seus elementos-filhos, que possuem mapas como seus elementos-filhos. Cada instância do mapa possui três estatísticas listadas. A estrutura mapModule é mostrada no diagrama a seguir:

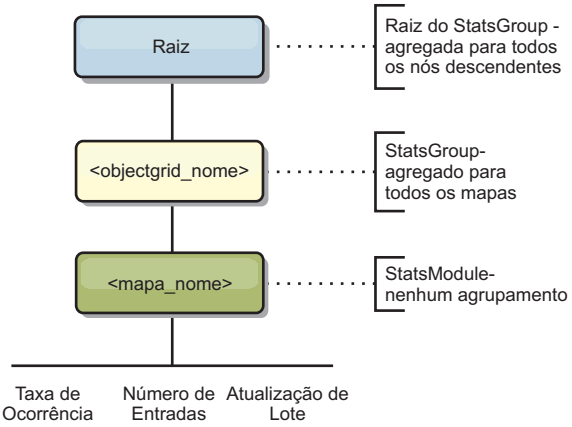
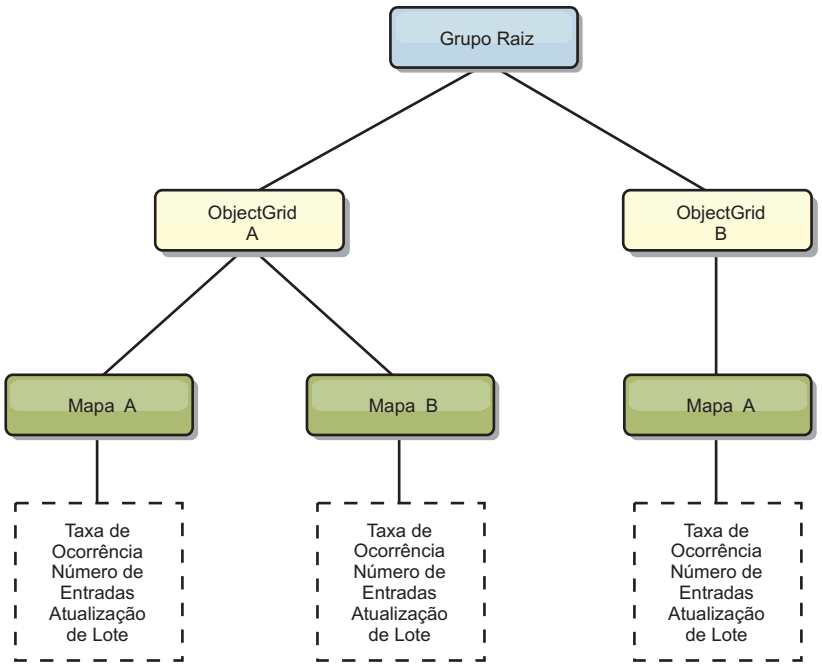


Figura 11. Estrutura do mapModule

O diagrama a seguir mostra um exemplo da estrutura mapModule:

Figura 12. Exemplo de Estrutura do Módulo mapModule



hashIndexModule

O hashIndexModule contém as seguintes estatísticas que são relacionadas aos índices de nível de Mapa:

- **Contagem de Localizações**-*CountStatistic*: O número de chamadas para a operação find do índice.
- **Contagem de Colisões**-*CountStatistic*: O número de colisões para a operação find.
- **Contagem de Falhas**-*CountStatistic*: O número de falhas para a operação find.
- **Contagem de Resultados**-*CountStatistic*: O número de chaves retornadas da operação find.
- **Contagem de BatchUpdate**-*CountStatistic*: O número de atualizações de lote junto a este índice. Quando o mapa correspondente é alterado de qualquer maneira, o método doBatchUpdate() do índice será chamado. Esta estatística informará com que frequência seu índice está sendo alterado ou atualizado.
- **Tempo de Duração da Operação Find**-*TimeStatistic*: A quantidade de tempo que a operação find leva para ser concluída

O elemento-raiz do hashIndexModule, "root", serve como ponto de entrada para as estatísticas do HashIndex. Este elemento-raiz possui ObjectGrids como seus elementos-filhos, os ObjectGrids possuem mapas como seus elementos-filhos, que, finalmente, possuem HashIndexes como seus elementos-filhos e nós-folhas da árvore. Cada instância do HashIndex possui três estatísticas listadas. A estrutura hashIndexModule é mostrada no diagrama a seguir:

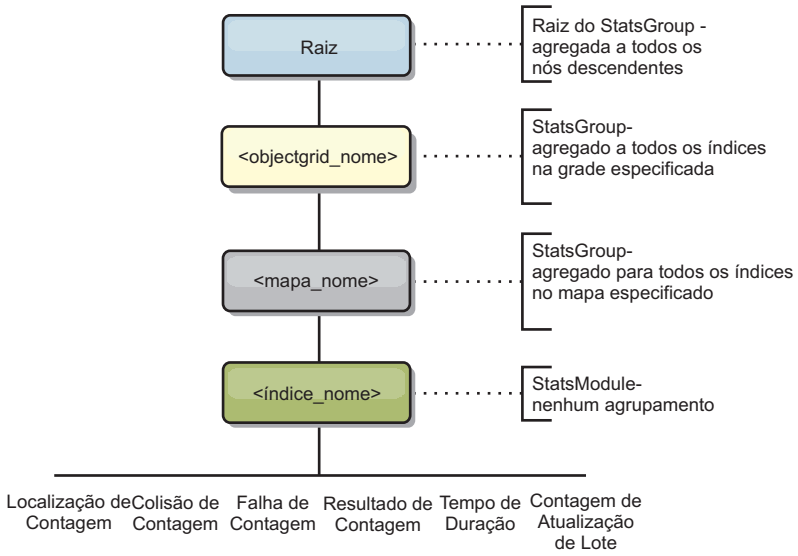


Figura 13. Estrutura do Módulo hashIndexModule

O diagrama a seguir mostra um exemplo da estrutura hashIndexModule:

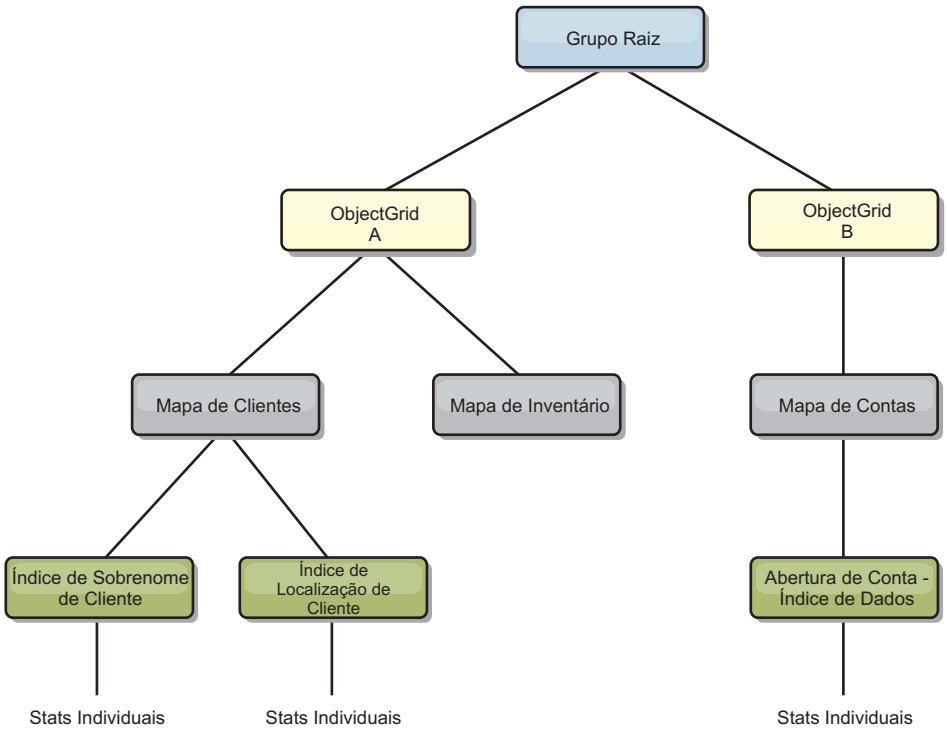


Figura 14. Exemplo da Estrutura do Módulo hashIndexModule

agentManagerModule

O agentManagerModule contém as estatísticas que são relacionadas aos agentes de nível de mapa:

- **Tempo de Redução**: *TimeStatistic* - A quantidade de tempo para o agente concluir a operação reduce.

- **Tempo de Duração Total**: *TimeStatistic* - A quantidade total de tempo para o agente concluir todas as operações.
- **Tempo de Serialização do Agente**: *TimeStatistic* - A quantidade de tempo para serialização do agente.
- **Tempo de Aumento do Agente**: *TimeStatistic* - A quantidade de tempo para aumentar o agente no servidor.
- **Tempo de Serialização do Resultado**: *TimeStatistic* - A quantidade de tempo para serializar os resultados do agente.
- **Tempo de Aumento do Resultado**: *TimeStatistic* - A quantidade de tempo para aumentar os resultados do agente.
- **Contagem de Falhas**: *CountStatistic* - O número de vezes que o agente falhou.
- **Contagem de Chamada**: *CountStatistic* - O número de vezes que o AgentManager foi chamado.
- **Contagem de Partições**: *CountStatistic* - O número de partições para as quais o agente é enviado.

O elemento-raiz do agentManagerModule, "root", serve como ponto de entrada para as estatísticas do AgentManager. O elemento-raiz possui ObjectGrids como seus elementos-filhos, os ObjectGrids possuem seus elementos-filhos, que, finalmente, possuem instâncias do AgentManager como seus elementos-filhos e nós-folhas da árvore. Cada instância do AgentManager possui três estatísticas listadas.

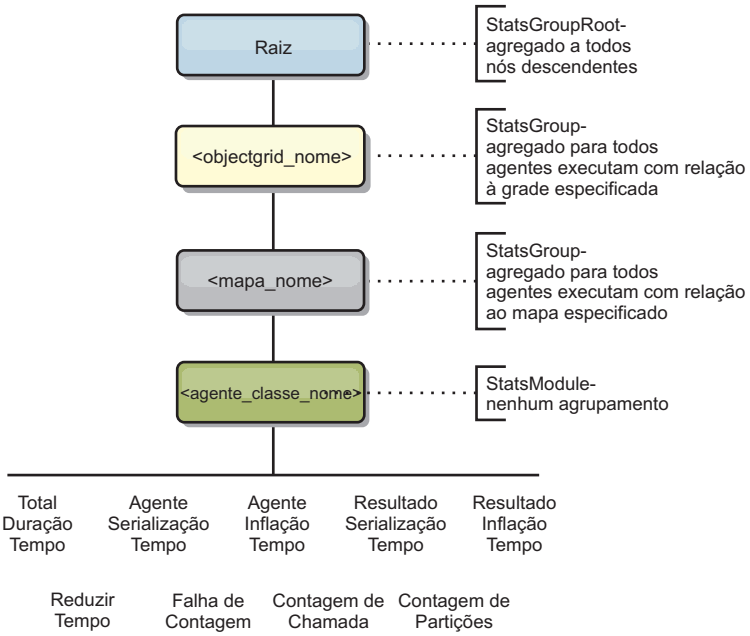


Figura 15. Estrutura agentManagerModule

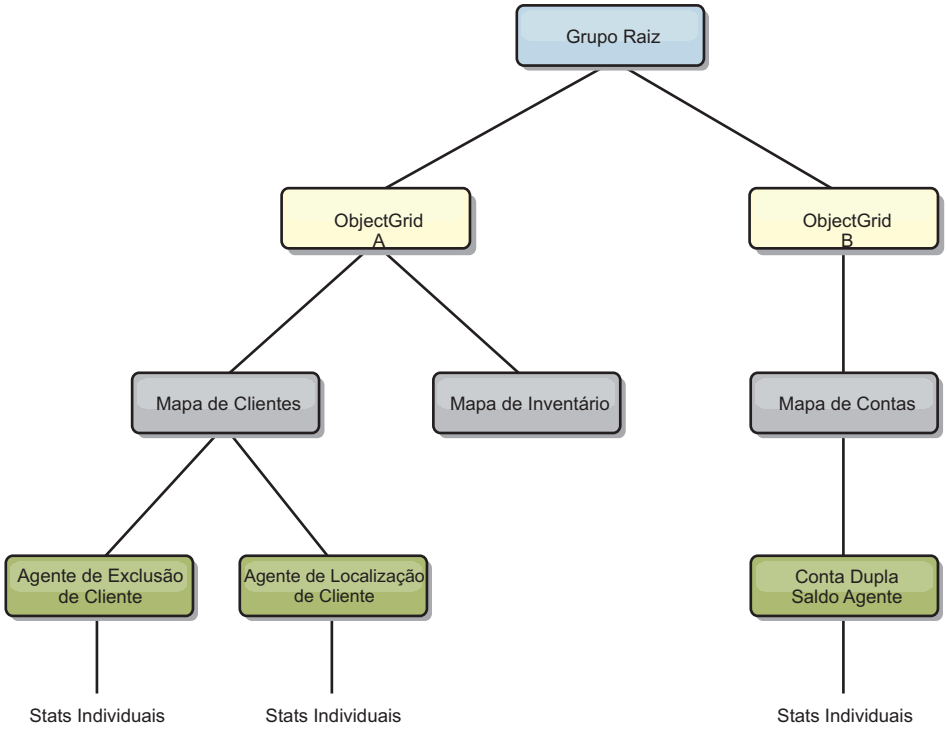


Figura 16. Exemplo da Estrutura agentManagerModule

queryModule

O queryModule contém estatísticas relacionadas às consultas do eXtreme Scale:

- **Tempo de Criação do Plano**: *TimeStatistic* - A quantidade de tempo para criar o plano de consulta.
- **Tempo de Execução**: *TimeStatistic* - A quantidade de tempo para executar a consulta.
- **Contagem de Execução**: *CountStatistic* - O número de vezes que a consulta foi executada.
- **Contagem de Resultados**: *CountStatistic* - A contagem para cada conjunto de resultados de cada execução de consulta.
- **FailureCount**: *CountStatistic* - O número de vezes que a consulta falhou.

O elemento-raiz do queryModule, "root", serve como ponto de entrada para as estatísticas do Query. Este elemento-raiz possui ObjectGrids como seus elementos-filhos, que possuem objetos Query como seus elementos-filhos e nós-folhas da árvore. Cada instância do Query possui as três estatísticas listadas.

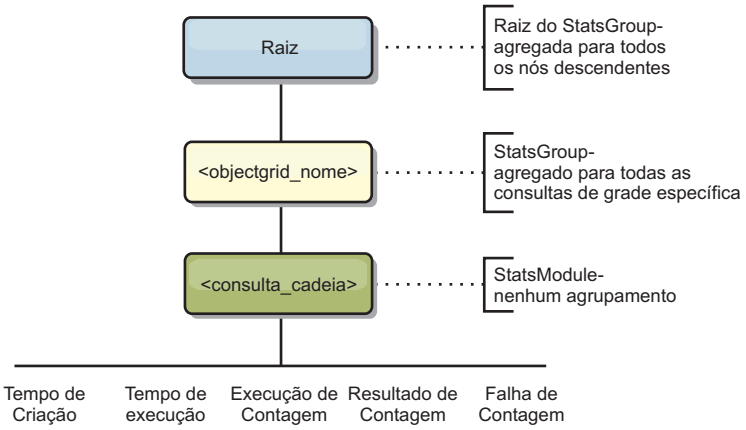


Figura 17. Estrutura queryModule

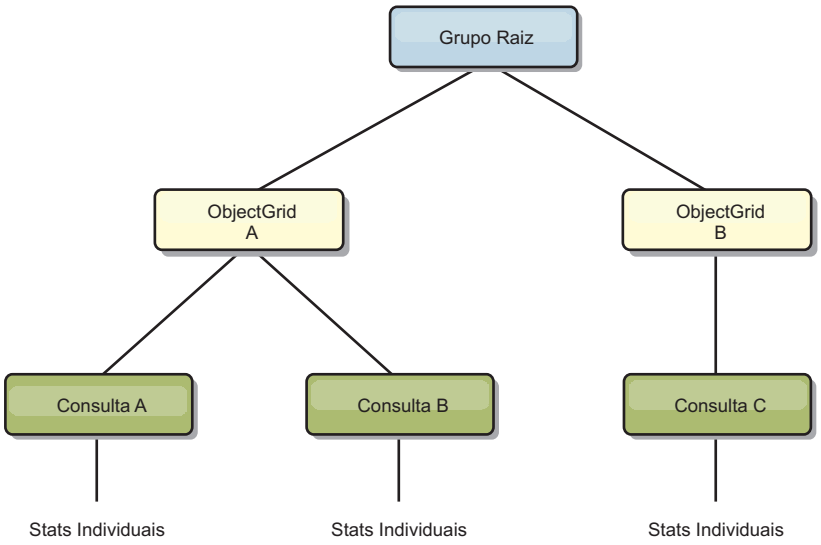


Figura 18. Exemplo da Estrutura queryModule QueryStats.jpg

Utilizando o Utilitário wsadmin

É possível utilizar o utilitário wsadmin fornecido no WebSphere Application Server para acessar as informações do MBean.

Acessando MBeans Utilizando a Ferramenta wsadmin

Execute a ferramenta wsadmin a partir do diretório bin em sua instalação do WebSphere Application Server. O exemplo a seguir recupera uma visualização da disposição do shard atual em um eXtreme Scale dinâmico. O wsadmin pode ser executado a partir de qualquer instalação na qual o eXtreme Scale está em execução. Não é necessário executar o wsadmin no serviço de catálogo.

```
$ wsadmin.sh -lang jython
wsadmin>placementService = AdminControl.queryNames
 ("com.ibm.websphere.objectgrid:*,type=PlacementService")
wsadmin>print AdminControl.invoke(placementService,
 "listObjectGridPlacement","library ms1")

<objectGrid name="library" mapSetName="ms1">
  <container name="container-0" zoneName="DefaultDomain"
  hostName="host1.company.org" serverName="server1">
     <shard type="Primary" partitionName="0"/>
     <shard type="SynchronousReplica" partitionName="1"/>
```

```
  </container>
  <container name="container-1" zoneName="DefaultDomain"
  hostName="host2.company.org" serverName="server2">
     <shard type="SynchronousReplica" partitionName="0"/>
     <shard type="Primary" partitionName="1"/>
  </container>
  <container name="UNASSIGNED" zoneName="_ibm_SYSTEM"
  hostName="UNASSIGNED" serverName="UNNAMED">
    <shard type="SynchronousReplica" partitionName="0"/>
    <shard type="AsynchronousReplica" partitionName="0"/>
  </container>
</objectGrid>
```

Utilizando beans gerenciados (MBeans) para administrar seu ambiente

É possível usar vários tipos diferentes de Java Management Extensions (JMX) MBeans para administrar e monitorar as implementações. Cada MBean faz referência a uma entidade específica, como um mapa, um eXtreme Scale, um servidor, um grupo de replicação ou um membro do grupo de replicação.

Interfaces MBean JMX e WebSphere eXtreme Scale

Cada MBean possui métodos get que representam valores de atributo. Estes métodos get não podem ser chamados diretamente a partir do seu programa. A especificação JMX trata os atributos de maneira diferente das operações. É possível visualizar atributos com um console JMX do fornecedor e executar operações em seu programa ou com um console JMX do fornecedor.

Visão Geral de Uso do MBean

É possível usar os beans gerenciados (MBeans) para controlar as estatísticas no seu ambiente.

Antes de Iniciar

Para que os atributos sejam registrados, é necessário ativar as estatísticas. É possível ativar as estatísticas de uma das seguintes formas:

- **Com o arquivo de propriedades do servidor:**

  É possível ativar as estatísticas no arquivo de propriedades do servidor com uma entrada de valor de chave `statsSpec=<StatsSpec>`. Alguns exemplos das possíveis configurações são:

  – Para ativar todas as estatísticas, use `statsSpec=all=enabled` ou `statisticsSpec=all=enabled`

  – Para ativar apenas as estatísticas do ObjectGrid, use `statsSpec=og.all=enabled`. Para visualizar uma descrição de todas as especificações de estatísticas possíveis, consulte o StatsSpec API na documentação da API.

  Para obter mais informações sobre o arquivo de propriedades do servidor, consulte “Arquivo de Propriedades do Servidor” na página 189.

- **Programaticamente:**

  Também é possível ativar as estatísticas programaticamente com a interface StatsAccessor, que é recuperada com a classe StatsAccessorFactory. Utilize esta interface em um ambiente do cliente ou quando precisar monitorar um eXtreme Scale que está em execução em outro processo.

Exemplo

Para obter um exemplo de como usar os beans gerenciados, consulte “Utilizando o Utilitário de Amostra xsAdmin” na página 266.

Utilizando o Utilitário de Amostra xsAdmin

Com o utilitário de amostra xsAdmin, é possível formar e exibir informações contextuais sobre sua topologia do WebSphere eXtreme Scale. O utilitário de amostra fornece um método para analisar e descobrir dados de implementação atuais e pode ser utilizado com uma base para criação de utilitários customizados.

Antes de Iniciar

É necessário ter o WebSphere eXtreme Scale instalado.

Por Que e Quando Desempenhar Esta Tarefa

É possível utilizar o utilitário de amostra xsAdmin para fornecer feedback sobre o layout atual e o estado específico da grade, tal como o conteúdo de mapa. Neste exemplo, o layout da grade nesta tarefa consiste em uma grade única, denominada *ObjectGridA* com um mapa definido, denominado *MapA*, pertencendo ao mapset, intitulado *MapSetA*. Este exemplo demonstra como é possível exibir todos os contêineres ativos em uma grade e imprimir métricas filtradas relativa ao tamanho do mapa do *MapA*. Para visualizar todas as opções de comando possíveis, execute o utilitário xsAdmin sem nenhum argumento ou com a opção **-help**.

1. Na linha de comandos, configure a variável de ambiente JAVA_HOME.
   - UNIX `export JAVA_HOME=javaHome`
   - Windows `set JAVA_HOME=javaHome`
2. Navegue até o diretório bin.
   ```
   cd objectGridRoot/bin
   ```
3. Ative o utilitário xsAdmin.
   - **Para exibir a ajuda on-line, execute o comando a seguir:** UNIX
     ```
     xsadmin.sh
     ```
     Windows
     ```
     xsadmin.bat
     ```
     Preste atenção à seção de argumentos necessários da mensagem de ajuda, porque é necessário passar apenas uma das opções listadas para o utilitário trabalhar. Se nenhuma opção **-g** ou **-m** for especificada, o utilitário xsAdmin imprime informações para cada grade na topologia.
   - **Para exibir todos os contêineres on-line para uma grade, execute o seguinte comando:** UNIX
     ```
     xsadmin.sh -g ObjectGridA -m MapSetA -containers
     ```
     Windows
     ```
     xsadmin.bat -g ObjectGridA -m MapSetA -containers
     ```
     Todas as informações do contêiner são exibidas. A seguir está um exemplo de saída:
     ```
      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
```

- **Para exibir o número de entradas em todos os mapas para uma grade, execute o seguinte comando:** UNIX

```
xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA
```

Windows

```
xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA
```

O tamanho do mapa especificado é exibido. A seguir está um exemplo de saída:

```
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
```

- **Para especificar a porta JMX para o serviço de catálogo, execute o seguinte comando:** O utilitário de amostra xsAdmin conecta-se ao servidor MBean que está em execução em um servidor de catálogos. Um servidor de catálogos pode ser executado em um processo independente, processo do WebSphere Application Server ou integrado em um processo aplicativo customizado. Utilize a opção **-ch** para especificar o nome do host do serviço de catálogo e a opção **-p** para especificar a porta de nomenclatura do serviço de catálogo.

UNIX

```
xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  -ch CatalogMachine -p 6645
```

Windows

```
xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA
  -ch CatalogMachine -p 6645
```

O tamanho do mapa especificado é exibido. A seguir está um exemplo de saída:

```
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
```

- **Para conectar-se a um serviço de catálogo hospedado em um processo do WebSphere Application Server, execute as seguintes etapas:**

A opção **-dmgr** é necessária ao conectar-se a um serviço de catálogo hospedado por qualquer processo do WebSphere Application Server ou cluster de processos. Utilize a opção **-ch** para especificar o nome do host se

não `localhost` e a opção **-p** para substituir a porta de autoinicialização do serviço de catálogo, que utiliza o processo BOOTSTRAP_ADDRESS. A opção **-p** é necessária apenas se o BOOTSTRAP_ADDRESS não estiver configurado com o padrão de 9809.

Nota: A versão independente do WebSphere eXtreme Scale não pode ser utilizada para conectar-se a um serviço de catálogo hospedado por um processo do WebSphere Application Server. Utilize o script xsAdmin incluído no diretório *was_root*`/bin`, que está disponível ao instalar o WebSphere eXtreme Scale no WebSphere Application Server ou WebSphere Application Server Network Deployment.

a. Navegue até o diretório bin do WebSphere Application Server:
```
cd wasRoot/bin
```

b. 2. Ative o utilitário xsAdmin utilizando o seguinte comando: UNIX
```
xsadmin.sh -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
```

Windows
```
xsadmin.bat -g ObjectGridA -m MapSetA -mapsizes -fm MapA -dmgr
```

c. O tamanho do mapa especificado é exibido.

```
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
```

Ferramentas de Fornecedor

O WebSphere eXtreme Scale pode ser monitorado usando diversas soluções populares de monitoramento corporativo. Os agentes do plug-in estão incluídos no IBM Tivoli Monitoring and Hyperic HQ, os quais monitoram o WebSphere eXtreme Scale usando os beans de gerenciamento acessíveis. O CA Wily Introscope usa a instrumentação do método Java para capturar estatísticas.

Monitorando com o IBM Tivoli Enterprise Monitoring Agent para WebSphere eXtreme Scale

O IBM Tivoli Enterprise Monitoring Agent é uma solução de monitoramento cheia de recursos que pode ser usada para monitorar bancos de dados, sistemas operacionais e servidores em ambientes distribuídos e de host. O WebSphere eXtreme Scale inclui um agente customizado que pode ser usado para fazer introspecção dos beans de gerenciamento do eXtreme Scale. Essa solução trabalha eficientemente para ambas as implementações do eXtreme Scale e do WebSphere Application Server independentes.

Antes de Iniciar

- Instale o WebSphere eXtreme Scale Versão 7.0.0 ou superior.
- Instale o IBM Tivoli Monitoring Versão 6.2.1 com o fix pack 2 ou superior.
- Instale o agente de S.O. Tivoli em cada servidor ou host no qual o servidor do eXtreme Scale é executado.

- Instale o agente do WebSphere eXtreme Scale, que pode ser obtido por download gratuitamente a partir do Site do IBM Open Process Automation Library (OPAL).

Conclua as seguintes etapas para instalar e configurar o Tivoli Monitoring Agent:

1. Instale o Tivoli Monitoring Agent para WebSphere eXtreme Scale.

   Faça download da imagem de instalação do Tivoli e extraia os arquivos para um diretório temporário.

2. Instale os arquivos de suporte do aplicativo eXtreme Scale.

   Instale o suporte do aplicativo eXtreme Scale em cada uma das seguintes implementações.

   - Tivoli Enterprise Portal Server (TEPS)
   - Enterprise Desktop client (TEPD)
   - Tivoli Enterprise Monitoring Server (TEMS)

   a. No diretório temporário que você criou, inicie uma nova janela de comando e execute o arquivo executável apropriado para sua plataforma. O script de instalação detecta automaticamente o tipo de implementação do Tivoli (TEMS, TEPD ou TEPS). É possível instalar qualquer tipo em um único host ou em vários hosts e todos os três tipos de implementação requerem a instalação dos arquivos de suporte de aplicativo do agente eXtreme Scale.

   b. Na janela do **Instalador**, verifique se as seleções para os componentes Tivoli implementados estão corretos. Clique em **Avançar**.

   c. Se for solicitado, envie o nome do host e as credenciais administrativas. Clique em **Avançar**.

   d. Selecione **Monitoring Agent para WebSphere eXtreme Scale**. Clique em **Avançar**.

   e. Você será notificado sobre quais ações de instalação deverão ser executadas. Clique em **Avançar** para visualizar o progresso da instalação até a conclusão.

   Depois de concluir o procedimento, todos os arquivos de suporte do aplicativo necessários pelo agente do WebSphere eXtreme Scale serão instalados.

3. Instale o agente em cada um dos nós do eXtreme Scale.

   Instale um agente do S.O. Tivoli em cada um dos computadores. Não é necessário configurar ou iniciar este agente. Use a mesma imagem de instalação da etapa anterior para executar o arquivo executável específico da plataforma.

   Como uma recomendação, você precisa instalar apenas um agente por host. Cada agente é capaz de suportar várias instâncias dos servidores eXtreme Scale. Para obter melhor desempenho, utiliza uma instância do agente para monitorar cerca de 50 servidores eXtreme Scale.

   a. Na tela de boas-vindas do assistente de instalação, clique em **Avançar** para abrir a tela para especificar as informações do caminho de instalação.

   b. Para o campo **Diretório de Instalação do Tivoli Monitoring**, insira ou procure por `C:\IBM\ITM` (ou `/opt/IBM/ITM`). Em seguida para o campo **Local para a Mídia de Instalação**, verifique se o valor exibido está correto e clique em **Avançar**.

   c. Selecione os componentes que deseja incluir, como **Executar uma instalação local da solução** e clique em **Avançar**.

   d. Selecione os aplicativos para os quais o suporte será incluído ao selecionar o aplicativo, como **Monitoring Agent para WebSphere eXtreme Scale** e clique em **Avançar**.

e. É possível visualizar o progresso até o suporte do aplicativo ser incluído com êxito.

Nota: Repita essas etapas em cada um dos nós do eXtreme Scale. Também é possível utilizar a instalação silenciosa. Consulte o IBM Tivoli Monitoring Information Center para obter mais informações sobre a instalação silenciosa.

4. Configure o agente do WebSphere eXtreme Scale.

   Cada um dos agentes instalado precisa ser configurado para monitorar qualquer servidor de catálogos, servidor eXtreme Scale ou ambos.

   As etapas para configurar as plataformas Windows eUNIX são diferentes. A configuração para a plataforma Windows é feita com a interface com o usuário **Gerenciar o Tivoli Monitoring Services**. A configuração para as plataformas UNIX baseia-se na linha de comandos.

   Windows Use as seguintes etapas para configurar inicialmente o agente no Windows Windows

   a. Na janela **Gerenciar o Tivoli Enterprise Monitoring Services**, clique em **Iniciar → Todos os Programas → IBM Tivoli Monitoring → Gerenciar o Tivoli Monitoring Services**.
   b. Clique com o botão direito do mouse em **Monitoring Agent para WebSphere eXtreme Scale** e selecione **Configurar usando padrão**, que abre uma janela para criar uma instância exclusiva do agente.
   c. Escolha um nome exclusivo, como por exemplo, `instance1` e clique em **Avançar**.

   • Windows Se planejar monitorar servidores do eXtreme Scale independentes, conclua as seguintes etapas:

     a. Atualize os parâmetros Java e certifique-se de que o valor **Java Home** esteja correto. Os argumentos da JVM podem ser deixados vazios. Clique em **Avançar**.
     b. Selecione o **Tipo de Conexão do Servidor MBean** e use `Servidor Compatível com JSR-160` para servidores eXtreme Scale independentes. Clique em **Avançar**.
     c. Se a segurança estiver ativada, atualize os valores de **ID do Usuário** e **Senha**. Deixe o valor **URL de Serviço JMX** como está. Substitua esse valor depois. Deixe o campo **Informações do Caminho da Classe JMX** como está. Clique em **Avançar**.

     Para configurar os servidores para o agente no Windows, conclua as seguintes etapas:

     a. Configure as instâncias de subnó dos servidores do eXtreme Scale na área de janela dos **Servidores de Grade do WebSphere eXtreme Scale**. Se nenhum servidor de contêiner existir no computador, clique em **Avançar** para continuar com a área de janela de serviço de catálogo.
     b. Se vários servidores de contêiner do eXtreme Scale existirem no computador, configure o agente para monitorar cada um dos servidores.
     c. É possível incluir quantos servidores do eXtreme Scale forem necessários se os nomes e portas forem exclusivos ao clicar em **Novo**. (Quando um servidor do eXtreme Scale for iniciado, um valor JMXPort deverá ser especificado).
     d. Depois de configurar os servidores de contêiner, clique em **Avançar**, para levá-lo para a área de janela **Servidores de Catálogo do WebSphere eXtreme Scale**.

e. Se você não possuir nenhum servidor de catálogo, clique em **OK**. Se você tiver servidores de catálogo, inclua uma nova configuração para cada servidor, conforme foi feito com os servidores de contêiner. Novamente, escolha um nome exclusivo, de preferência o mesmo nome que foi usado ao iniciar o serviço de catálogo. Clique **OK** para concluir.

- Windows Se você planejar monitorar servidores para o agente nos servidores do eXtreme Scale que são integrados em um processo do WebSphere Application Server, conclua as seguintes etapas:
  a. Atualize os parâmetros Java e certifique-se de que o valor **Java Home** esteja correto. Os argumentos da JVM podem ser deixados vazios. Clique em **Avançar**.
  b. Selecione o **Tipo de Conexão do Servidor MBean**. Selecione a versão do WebSphere Application Server que seja apropriada para seu ambiente. Clique em **Avançar**.
  c. Certifique-se de que as informações do WebSphere Application Server no painel estejam corretas. Clique em **Avançar**.
  d. Inclua apenas uma definição de subnó. Forneça um nome para a definição de subnó, mas não atualize a definição de porta. No ambiente do WebSphere Application Server, os dados podem ser coletados de todos os servidores de aplicativos que forem gerenciados pelo agente do nó que estiver em execução no computador. Clique em **Avançar**.
  e. Se nenhum servidor de catálogo existir no ambiente, clique em **OK**. Se você tiver servidores de catálogo, inclua uma nova configuração para cada servidor, conforme foi feito com os servidores de contêiner. Escolha um nome exclusivo para o serviço de catálogo, de preferência o mesmo nome que foi usado ao iniciar o serviço de catálogo. Clique **OK** para concluir.

Nota: Os servidores de contêineres não precisam ser colocados junto com o serviço de catálogo.

Agora que o agente e os servidores estão configurados e prontos, na próxima janela, clique com o botão direito do mouse em `instance1` para iniciar o agente.

UNIX Para configurar o agente na plataforma UNIX na linha de comandos, conclua as seguintes etapas:

A seguir há um exemplo de servidores independentes que usam um tipo de conexão Compatível com JSR160. O exemplo mostra três contêineres do eXtreme Scale em um único host (rhea00b02) e os endereços de listener JMX são 15000, 15001 e 15002, respectivamente. Não há nenhum servidor de catálogo.

A saída do utilitário de configuração é exibida em *`itálico de espaço simples`*, enquanto que a resposta do usuário está em **`negrito de espaço simples`**. (Se nenhuma resposta do usuário era necessária, o padrão foi selecionado pressionando a tecla enter). UNIX

```
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...
```

O exemplo anterior cria uma instância do agente chamada “inst1” e atualiza as configurações Java Home. Os servidores de contêiner do eXtreme Scale são configurados, mas o serviço de catálogo não é configurado.

Nota: O procedimento anterior cria um arquivo de texto no seguinte formato no diretório: <ITM_install>/config/<host>_xt_<instance name>.cfg.

Exemplo: rhea00b02_xt_inst1.cfg

É recomendado editar esse arquivo com seu editor de texto simples escolhido. A seguir há um exemplo de conteúdo desse arquivo:

```
INSTANCE=inst2 [        SECTION=KQZ_JAVA [ { JAVA_HOME=/opt/OG61/java }  { JAVA_TRACE_LEVEL=ERROR }  ]
SECTION=KQZ_JMX_CONNECTION_SECTION [ { KQZ_JMX_CONNECTION_PROPERTY=KQZ_JMX_JSR160_JSR160 }  ]
SECTION=KQZ_JMX_JSR160_JSR160 [ { KQZ_JMX_JSR160_JSR160_CLASS_PATH_TITLE= }
{ KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:rmi:///jndi/rmi://localho
st:port/objectgrid/MBeanServer }  { KQZ_JMX_JSR160_JSR160_CLASS_PATH_SEPARATOR= }  ]
SECTION=OGS:rhea00b02_c1 [ { KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:
rmi:///jndi/rmi://localhost:15001/objectgrid/MBeanServer }  ]
SECTION=OGS:rhea00b02_c0 [ { KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:
rmi:///jndi/rmi://localhost:15002/objectgrid/MBeanServer }  ]
SECTION=OGS:rhea00b02_c2 [ { KQZ_JMX_JSR160_JSR160_SERVICE_URL=service:jmx:
rmi:///jndi/rmi://localhost:15002/objectgrid/MBeanServer }  ]]
```

A seguir há um exemplo que mostra uma configuração de uma implementação do WebSphere Application Server:

```
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 #
```

Para implementações do WebSphere Application Server, não é necessário criar vários subnós. O agente do eXtreme Scale se conecta como agente de nó para reunir todas as informações a partir dos servidores de aplicativos para os quais ele é responsável.

`SECTION=CAT` significa uma linha de serviço de catálogo, enquanto `SECTION=OGS` significa uma linha de configuração do servidor eXtreme Scale.

5. Configure os servidores do eXtreme Scale.

   Quando os servidores de contêiner do eXtreme Scale forem iniciados sem especificar o argumento **-JMXServicePort**, um servidor MBean será designado a uma porta dinâmica. O agente precisa saber antecipadamente com qual porta JMX ele se comunicará. O agente não trabalha com portas dinâmicas.

   Ao iniciar os servidores, é necessário especificar o argumento `-JMXServicePort <port_number>` ao iniciar o servidor eXtreme Scale usando o comando `startOgServer.sh | .bat`. Executar esse comando garante que o servidor JMX dentro do processo atenda em uma porta estática predefinida.

   Para os exemplos anteriores em uma instalação UNIX, dois servidores eXtreme Scale precisarão ser iniciados com as portas configuradas:

   a. ″-JMXServicePort″ ″15000″ (para rhea00b02_c0)

   b. ″-JMXServicePort″ ″15001″ (para rhea00b02_c1)

   a. Inicie o agente WebSphere eXtreme Scale.

      Supondo que a instância `inst1` foi criada, como no exemplo anterior, emita os seguintes comandos.

      1) `cd <ITM_install>/bin`

      2) `itmcmd agent -o inst1 start xt`

   b. Pare o agente do WebSphere eXtreme Scale.

      Supondo que a instância “inst1” foi criada, como no exemplo anterior, emita os seguintes comandos.

      1) `cd <ITM_install>/bin`

```
2) itmcmd agent -o inst1 stop xt
```

Resultados

Após todos os servidores serem configurados e iniciados, os dados do MBeans são exibidos no console do IBM Tivoli Portal. As áreas de trabalho predefinidas mostram gráficos e métricas de dados em cada nível de nó.

As seguintes áreas de trabalho são definidas: Nó **Servidores de Grade do WebSphere eXtreme Scale** para todos os nós monitorados.

- Visualização Transações do eXtreme Scale
- Visualização Shard Primário do eXtreme Scale
- Visualização Memória do eXtreme Scale
- Visualização ObjectMap do eXtreme Scale

Também é possível configurar suas próprias áreas de trabalho. Para obter mais informações, consulte as informações sobre a customização das áreas de trabalho no IBM Tivoli Monitoring Information Center.

Monitorando Aplicativos eXtreme Scale com o CA Wily Introscope

O CA Wily Introscope é um produto de gerenciamento de terceiro que pode ser usado para detectar e diagnosticar problemas de desempenho em ambientes de aplicativos corporativos. O eXtreme Scale inclui detalhes sobre a configuração do CA Wily Introscope para fazer introspecção de partes selecionadas do tempo de execução do eXtreme Scale para visualizar e validar rapidamente os aplicativos do eXtreme Scale. O CA Wily Introscope funciona eficientemente para as implementações do WebSphere Application Server e independentes.

Visão Geral

Para monitorar aplicativos eXtreme Scale com o CA Wily Introscope, é necessário colocar as configurações nos arquivos ProbeBuilderDirective (PBD) que fornecem acesso às informações de monitoramento para o eXtreme Scale.

Atenção: Os pontos de instrumentação para o Introscope podem ser alterados com cada fix pack ou release. Ao instalar um novo fix pack ou release, consulte a documentação para saber se há alguma alteração nos pontos de instrumentação.

é possível configurar os arquivos ProbeBuilderDirective (PBD) do CA Wily Introscope para monitorar seus aplicativos eXtreme Scale. O CA Wily Introscope é um produto de gerenciamento de aplicativos com o qual você pode detectar, selecionar e diagnosticar proativamente problemas de desempenho nos seus ambientes complexos, compostos e de aplicativos da Web.

Configurações de Arquivos PBD para Monitoramento do Serviço de Catálogo

É possível utilizar uma ou mais das seguintes configurações em seu arquivo PBD para monitorar o serviço de catálogo.

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  changeDefinedCompleted
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  viewChangeCompleted
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
```

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  viewAboutToChange
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeat
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatCluster
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatCurrentLeader
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatDeadServer
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatNewLeader
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatNewServer
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  PlacementServiceImpl
importRouteInfo BlamePointTracerDifferentMethods
  "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  PlacementServiceImpl heartbeat
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  PlacementServiceImpl joinPlacementGroup
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
  TraceOneMethodOfClass:
com.ibm.ws.objectgrid.catalog.placement.PlacementServiceImpl
  classifyServer
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  BalanceGridEventListener shardActivated
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.catalog.placement.
  BalanceGridEventListener shardDeactivate
BlamePointTracerDifferentMethods "OGcatalog|{classname}|{method}"
```

Classes para monitoramento do serviço de catálogo

HAControllerImpl
A classe HAControllerImpl manipula eventos de ciclo de vida e feedback do grupo principal. É possível monitorar esta classe para obter uma indicação da estrutura e das alterações do grupo principal.

ServerAgent
A classe ServerAgent é responsável pela comunicação de eventos do grupo principal com o serviço de catálogo. É possível monitorar as diversas chamadas de pulsação para marcar eventos graves.

PlacementServiceImpl
A classe PlacementServiceImpl coordena os contêineres. É possível utilizar os métodos nesta classe para monitorar eventos de junção e disposição do servidor.

BalanceGridEventListener
A classe BalanceGridEventListener controla a liderança de catálogo. É possível monitorar esta classe para obter uma indicação de qual serviço de catálogo está atualmente atuando como o líder.

Configurações de Arquivos PBD para Monitoramento de Contêineres

É possível utilizar uma ou mais das seguintes configurações em seu arquivo PBD para monitorar os contêineres.

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.ShardImpl processMessage
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.plugins.
  CommittedLogSequenceListenerProxy applyCommitted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.plugins.
  CommittedLogSequenceListenerProxy sendApplyCommitted
```

```
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.map.BaseMap evictMapEntries
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.checkpoint.
  CheckpointMapImpl$CheckpointIterator activateListener
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  changeDefinedCompleted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  viewChangeCompleted
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.hamanager.HAControllerImpl
  viewAboutToChange
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  batchProcess
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeat
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatCluster
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatCurrentLeader
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatDeadServer
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatNewLeader
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.container.ServerAgent
  heartbeatNewServer
BlamePointTracerDifferentMethods "OGcontainer|{classname}|{method}"
```

Classes para monitoramento dos contêineres

ShardImpl

A classe ShardImpl possui o método processMessage. O método processMessage é o método para pedidos do cliente. Com este método, é possível obter contagens de tempo de resposta e de pedidos do lado do servidor. Ao observar as contas em todos os servidores e monitorar a utilização do heap, é possível determinar se a grade está equilibrada.

CheckpointIterator

A classe CheckpointIterator possui a chamada de método activateListener que coloca primários no modo peer. Quando os primárias são colocadas no modo peer, a réplica é atualizada com o primário após a conclusão do método. Quando uma réplica está se regenerando a partir de um primário completo, esta operação pode levar um período estendido de tempo. O sistema não é totalmente recuperado até a conclusão da operação, portanto, você pode utilizar esta classe para monitorar o progresso da operação.

CommittedLogSequenceListenerProxy

A classe CommittedLogSequenceListenerProxy possui dois métodos de interesse. O método applyCommitted é executado para cada transação e o sendApplyCommitted é executado enquanto a réplica está executando pull de informações. A proporção de com que frequência estes dois métodos são executados pode fornecer a você alguma indicação de quão bem a réplica é capaz de continuar com o primário.

Configurações de Arquivos PBD para Monitoramento dos Clientes

É possível utilizar uma ou mais das seguintes configurações em seu arquivo PBD para monitorar os clientes.

```
TraceOneMethodOfClass: com.ibm.ws.objectgrid.client.ORBClientCoreMessageHandler
 sendMessage
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.corba.cluster.ClusterStore
  bootstrap
```

```
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.corba.cluster.ClusterStore
  epochChangeBootstrap
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.map.BaseMap evictMapEntries
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.cluster.orb.routing.
  SelectionServiceImpl routeFailed
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.cluster.orb.routing.
  SelectionServiceImpl routeFailed
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.SessionImpl getMap
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TraceOneMethodOfClass: com.ibm.ws.objectgrid.ObjectGridImpl getSession
BlamePointTracerDifferentMethods "OGclient|{classname}|{method}"
TurnOn: ObjectMap
SetFlag: ObjectMap
IdentifyClassAs: com.ibm.ws.objectgrid.ObjectMapImpl ObjectMap
TraceComplexMethodsifFlagged: ObjectMap BlamePointTracerDifferentMethods
"OGclient|{classname}|{method}"
```

Classes para monitoramento de clientes

ORBClientCoreMessageHandler
A classe ORBClientCoreMessageHandler é responsável por enviar pedidos de aplicativos para os contêineres. É possível monitorar o método sendMessage para o tempo de resposta do cliente e o número de pedidos.

ClusterStore
A classe ClusterStore contém as informações de roteamento no lado do cliente.

BaseMap
A classe BaseMap possui o método evictMapEntries que é chamado quando o evictor deseja remover entradas do mapa.

SelectionServiceImpl
A classe SelectionServiceImpl toma as decisões de roteamento. Se o cliente está tomando decisões de failover, é possível utilizar esta classe para ver as ações que são concluídas a partir das decisões.

ObjectGridImpl
A classe ObjectGridImpl possui o método getSession que você pode monitorar para ver o número de pedidos para este método.

Monitorando o eXtreme Scale com o Hyperic HQ

O Hyperic HQ é uma solução de monitoramento de terceiro que está disponível gratuitamente como uma solução de software livre ou como um produto corporativo. O WebSphere eXtreme Scale inclui um plug-in que permite que os agentes do Hyperic HQ descubram os servidores de contêiner do eXtreme Scale e relatem e agreguem as estatísticas usando os beans de gerenciamento do eXtreme Scale. É possível usar o Hyperic HQ para monitorar as implementações do eXtreme Scale independentes.

Antes de Iniciar

- Esse conjunto de instruções destina-se para o Hyperic Versão 4.0. Se você tiver uma versão mais recente do Hyperic, consulte a Documentação do Hyperic para obter informações sobre os nomes de caminho e como iniciar os agentes e servidores.
- Download do servidor Hyperic e das instalações do agente. Uma instalação do servidor deve estar em execução. Para detectar todos os servidoreseXtreme Scale, um agente Hyperic deve estar em execução em cada máquina na qual um servidor eXtreme Scale está em execução. Consulte o Web site Hyperic para obter informações de download e suporte para documentação.

- É necessário ter acesso aos arquivos `objectgrid-plugin.xml` e `hqplugin.jar`. Esses arquivos estão no diretório `objectgridRoot/hyperic/etc`.

Por Que e Quando Desempenhar Esta Tarefa

Ao integrar o eXtreme Scale com o software de monitoramento Hyperic HQ, poderá monitorar e exibir graficamente as métricas sobre o desempenho do seu ambiente. Configure essa integração ao usar uma implementação de plug-in em cada agente.

1. Inicie os servidores eXtreme Scale. O plug-in Hyperic procura nos processos locais para se conectar ao Java Virtual Machines que está executando o eXtreme Scale. Para se conectar corretamente com o Java Virtual Machines, cada servidor deve ser iniciado com a opção **-jmxServicePort**. Também é necessário ativar as estatísticas no arquivo de propriedades do servidor. Os servidores de catálogo não são detectados por esse filtro.
   - Para obter informações sobre como iniciar os servidores com a opção **-jmxServicePort**, consulte "Script startOgServer" na página 223.
   - Para obter informações sobre a ativação das estatísticas no arquivo de propriedades do servidor, consulte as informações sobre a propriedade **statsSpec** no "Arquivo de Propriedades do Servidor" na página 189.
2. Coloque os arquivos `extremescale-plugin.xml` e `wxshyperic.jar` nos diretórios de plug-in do servidor e do agente apropriados na configuração Hyperic. Para integração com o Hyperic, ambas as instalações do agente e do servidor devem ter acesso ao plug-in e aos arquivos Java archive (JAR). Embora o servidor possa trocar as configurações dinamicamente, é necessário concluir a integração antes de iniciar qualquer um dos agentes.
   a. Coloque o arquivo `extremescale-plugin.xml` no diretório `plugin` do servidor, que está no seguinte local:
      `hyperic_home/server_home/hq-engine/server/default/deploy/hq.ear/hq-plugins`
   b. Coloque o arquivo `extremescale-plugin.xml` no diretório `plugin` do agente, que está no seguinte local:
      `agent_home/bundles/gent-4.0.2-939/pdk/plugins`
   c. Coloque o arquivo `wshyperic.jar` no diretório `lib` do agente, que está no seguinte local:
      `agent_home/bundles/gent-4.0.2-939/pdk/lib`
3. Configure o agente. O arquivo `agent.properties` atua como um ponto de configuração para o tempo de execução do Agente. Essa propriedade está no diretório *agent_home*`/conf`. As seguintes chaves são opcionais, mas importantes para o plug-in do eXtreme Scale:
   -
     `autoinventory.defaultScan.interval.millis=<`*time_in_milliseconds*`>`

     Configura o intervalo em milissegundos entre as descobertas do Agente.
   -
     `log4j.logger.org.hyperic.hq.plugin.extremescale.XSServerDetector=DEBUG`

     : Ativa as instruções de depuração detalhadas a partir do plug-in eXtreme Scale.
   - `username=<`*username*`>`: Configura o nome do usuário do Java Management Extensions (JMX) se a segurança estiver ativada.
   - `password=<`*password*`>`: Configura a senha JMX se a segurança estiver ativada.
   - `sslEnabled=<`*true*`|`*false*`>`: Informa se o plug-in deve usar ou não o Secure Sockets Layer (SSL). Por padrão, o valor é `false`.

- `trustPath=<path>`: Configura um caminho de confiança para a conexão SSL.
- `trustType=<type>`: Configura um tipo de confiança para a conexão SSL.
- `trustPass=<password>`: Configura uma senha de confiança para a conexão SSL.

4. Inicie a descoberta do agente. Os agentes Hyperic enviam informações de descobertas e de métricas para o servidor. Utilize o servidor para customizar visualizações de dados e de objetos de inventário lógicos do grupo para gerar informações úteis. Depois que o servidor estiver disponível, é necessário executar o script de ativação ou iniciar o serviço do Windows para o agente:
   - Linux `agent_home/bin/hq-agent.sh start`
   - Windows Inicie o agente com o serviço do Windows.

   Depois de iniciar os agentes, os servidores são detectados e os grupos configurados. É possível efetuar login no console do servidor e escolher quais recursos incluir no banco de dados de inventário para o servidor. Por padrão, o console do servidor está no seguinte URL: `http://<server_host_name>:7080/`
5. Monitorar servidores com o console do Hyperic. Depois que os servidores forem incluídos no modelo de inventário, os serviços não serão mais necessários.
   - **Visualização Painel**: Quando você visualizou os eventos de detecção de recursos, efetuou login na visualização de painel principal. A visualização de painel é uma visualização genérica que atua como um centro de mensagens que pode ser customizado. É possível exportar gráficos ou objetos de inventário para este painel principal.
   - **Visualização Recursos**: É possível consultar e visualizar o modelo de inventário inteiro a partir desta página. Após os serviços terem sido incluídos, é possível visualizar todos os servidores eXtreme Scale adequadamente rotulados e listados juntos sob a seção Servidores. É possível clicar nos servidores individuais para visualizar as métricas básicas.
6. Visualize o inventário do servidor inteiro na página Visualização de Recursos. Nesta página, você poderá selecionar vários servidores ObjectGrid e agrupá-los. Depois de agrupar um conjunto de recursos, as métricas comuns podem ser exibidas em gráfico para mostrar as sobreposições e diferenças entre os membros de grupo. Para exibir uma sobreposição, selecione as métricas na exibição do Grupo de Servidor. Em seguida, a métrica é exibida na área de gráfico. Para exibir uma sobreposição para todos os membros de grupo, clique no nome da métrica sublinhada. É possível exportar qualquer gráfico, visualizações de nó e sobreposições comparativas no painel principal com o menu **Ferramentas**.

Logs e Rastreio

É possível utilizar logs e rastreio para monitorar e solucionar problemas em seu ambiente. Os logs estão em locais diferentes dependendo da sua configuração. Pode ser necessário fornecer rastreio para um servidor quando você trabalha com o suporte IBM.

Logs com o WebSphere Application Server

Consulte o WebSphere Application ServerCentro de Informações para obter mais informações.

Logs com o WebSphere eXtreme Scale em um Ambiente Independente

Com os servidores de catálogos e contêineres independentes, é possível configurar o local dos logs e qualquer especificação de rastreio. Os logs do servidor de catálogo estão no local onde você executou o comando do servidor de início.

Configurando o local de log para os servidores de contêiner

Por padrão, os logs para um contêiner estão no diretório onde o comando do servidor foi executado. Se você iniciar os servidores no diretório *<eXtremeScale_home>*/bin, os arquivos de log e de rasteio estão nos diretórios logs/*<server_name>* no diretório bin. Para especificar um local alternativo para os logs do servidor de contêineres, como o arquivo server.properties, com os seguintes conteúdos:

```
workingDirectory=<directory>
traceSpec=
systemStreamToFileEnabled=true
```

A propriedade workingDirectory é o diretório-raiz para os logs e o arquivo de rastreio opcional. O WebSphere eXtreme Scale cria um diretório com o nome do servidor de contêineres com um arquivo SystemOut.log, um arquivo SystemErr.log e um arquivo de rastreio se o rastreio foi ativado com a opção traceSpec. Para usar um arquivo de propriedades durante a inicialização de contêiner, use a opção **-serverProps** e forneça o local do arquivo de propriedades do servidor.

Consulte "Iniciando Servidores WebSphere eXtreme Scale Independentes" na página 218 e "Script startOgServer" na página 223 para obter informações adicionais.

As mensagens de informações comuns a serem procuradas no arquivo SystemOut.log são o início das mensagens de confirmação. Para obter mais informações sobre uma mensagem específica, consulte "Mensagens" na página 309.

Rastreio com o WebSphere Application Server

Consulte o WebSphere Application ServerCentro de Informações para obter mais informações.

Rastreio no Serviço de Catálogo

É possível configurar o rastreio em um serviço de catálogo usando os parâmetros **-traceSpec** e **-traceFile** durante a inicialização do serviço de catálogo. Por exemplo:

```
startOgServer.sh catalogServer -traceSpec
ObjectGridPlacement=all=enabled -traceFile
/home/user1/logs/trace.log
```

Se você iniciar o serviço de catálogo no diretório *<eXtremeScale_home>*/bin, os arquivos de log e de rasteio estarão no diretório logs/*<catalog_service_name>* no diretório bin. Consulte o "Iniciando o Serviço de Catálogo em um Ambiente Independente" na página 218 para obter mais detalhes sobre o início de um serviço de catálogo.

Rastreio em um servidor de contêineres Independente

É possível ativar o rastreio em um servidor de contêiner de duas formas. É possível criar um arquivo de propriedades do servidor conforme explicado na seção de logs ou você pode ativar o rastreio utilizando a linha de comandos na

inicialização. Para ativar o rastreio de contêiner com um arquivo de propriedades do servidor, atualize a propriedade **traceSpec** com a especificação de rastreio necessária. Para ativar o rastreio de contêiner utilizando parâmetros de início, utilize os parâmetros **-traceSpec** e **-traceFile**. Por exemplo:

```
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
```

Se você iniciar o servidor no diretório *<eXtremeScale_home>*/bin, os arquivos de log e de rasteio estarão nos diretórios logs/*<server_name>* no diretório bin. Consulte o "Iniciando Processos do Contêiner" na página 221 para obter mais detalhes sobre o início de um processo de contêiner.

Rastreio com a Interface ObjectGridManager

Outra opção é configurar o rastreio durante o tempo de execução em uma interface ObjectGridManager. A configuração de um rastreio em uma interface ObjectGridManager pode ser usada para obter rastreio em um cliente eXtreme Scale enquanto ele se conecta com um eXtreme Scale e confirma as transações. Para configurar o rastreio em uma interface ObjectGridManager, forneça uma especificação de rastreio e um log de rastreio.

```
ObjectGridManager manager= ObjectGridManagerFactory.getObjectGridManager();
...
manager.setTraceEnabled(true);
manager.setTraceFileName("logs/myClient.log");
manager.setTraceSpecification("ObjectGridReplication=all=enabled");
```

Ativando o Rastreio com o Utilitário xsadmin

Para ativar o rastreio com o utilitário xsadmin, use a opção **setTraceSpec**. Use o utilitário xsadmin para ativar o rastreio em um ambiente independente durante o tempo de execução e não durante a inicialização. É possível ativar o rastreio em todos os servidores e serviços de catálogo ou filtrar os servidores com base no nome do ObjectGrid, e assim por diante. Por exemplo, para ativar o rastreio ObjectGridReplication com acesso ao servidor de serviço de catálogo, execute:

```
<eXtremeScale_home>/bin>xsadmin.bat -setTraceSpec "ObjectGridReplication=all=enabled"
```

Também é possível desativar o rastreio ao configurar a especificação de rastreio para `*=all=disabled.`:

Consulte o "Utilizando o Utilitário de Amostra xsAdmin" na página 266 para obter mais informações.

Diretório e Arquivos ffdc

Os arquivos FFDC servem para que o suporte IBM auxilie na depuração. Estes arquivos podem ser solicitados pelo suporte IBM se ocorrer um problema.

Esses arquivos aparecem no diretório ffdc e contêm arquivos semelhantes ao seguinte:

```
server2_exception.log
server2_20802080_07.03.05_10.52.18_0.txt
```

Opções de Rastreio

É possível ativar o rastreio para fornecer informações sobre o seu ambiente para o suporte IBM.

Sobre o Rastreio

O rastreio do WebSphere eXtreme Scale é dividido em vários componentes diferentes. Assim como o rastreio do WebSphere Application Server, é possível especificar o nível de rastreio a ser utilizado. Os níveis comuns de rastreio incluem: all, debug, entryExit e event.

Um exemplo de cadeia de rastreio é o seguinte:

```
ObjectGridComponent=level=enabled
```

É possível concatenar as cadeias de rastreio. Use o sinal de asterisco (*) para especificar um valor de curinga, como `ObjectGrid*=all=enabled`. Se for necessário fornecer um rastreio para o suporte IBM, uma cadeia de rastreio específica será solicitada. Por exemplo, se ocorrer um problema com a replicação, a cadeia de rastreio `ObjectGridReplication=debug=enabled` pode ser solicitada.

Especificação de Rastreio

ObjectGrid
Mecanismo de cache principal geral.

ObjectGridCatalogServer
Serviço de catálogo geral.

ObjectGridChannel
Comunicações de topologia de implementação estática.

ObjectgridCORBA
Comunicações de topologia de implementação dinâmica.

ObjectGridDataGrid
A API do AgentManager.

ObjectGridDynaCache
O provedor de cache dinâmico do WebSphere eXtreme Scale.

ObjectGridEntityManager
A API do EntityManager. Utilize com a opção Projector.

ObjectGridEvictors
Evictors integrados do ObjectGrid.

ObjectGridJPA
Carregadores do Java Persistence API (JPA).

ObjectGridJPACache
Plug-ins do Cache JPA

ObjectGridLocking
Gerenciador de bloqueios de entrada de cache do ObjectGrid.

ObjectGridMBean
Beans de gerenciamento.

ObjectGridPlacement
Serviço de disposição de shards do servidor de catálogos.

ObjectGridQuery
Consulte ObjectGrid.

ObjectGridReplication
Serviço de replicação.

ObjectGridRouting
Detalhes de roteamento do cliente/servidor.

ObjectGridSecurity
Rastreio de segurança.

ObjectGridStats
Estatísticas do ObjectGrid.

ObjectGridStreamQuery
API de Consulta do Fluxo.

ObjectGridWriteBehind
Atributo write-behind do ObjectGrid.

Projector
O mecanismo com a API do EntityManager.

QueryEngine
O mecanismo de consulta para a API de Consulta do Objeto e a API de Consulta do EntityManager.

QueryEnginePlan
Diagnósticos do plano de consulta.

Capítulo 9. Protegendo o Ambiente de Implementação

Para proteger seus dados do WebSphere eXtreme Scale, o eXtreme Scale pode integrar-se com os provedores de segurança externos.

O WebSphere eXtreme Scale pode integra-se a uma implementação de segurança externa. Esta implementação externa deve fornecer serviços de autenticação e autorização para o eXtreme Scale. O eXtreme Scale possui pontos de plug-in para integrar-se com uma implementação de segurança. O eXtreme Scale foi integrado com êxito com os seguintes componentes:

- LDAP (Lightweight Directory Access Protocol)
- Kerberos
- Segurança do ObjectGrid
- Tivoli Access Manager
- JAAS (Java Authentication and Authorization Service)

O eXtreme Scale utiliza o provedor de segurança para as seguintes tarefas:

- Autenticar clientes para servidores.
- Autorizar clientes para acessar determinados artefatos do eXtreme Scale ou para especificar o que pode ser feito com os artefatos do eXtreme Scale.

O eXtreme Scale possui os seguintes tipos de autorizações:

Autorização de mapa
: Clientes ou grupos podem ser autorizados a executar operações de inserção, leitura, atualização, despejo ou exclusão nos mapas.

Autorização do ObjectGrid
: Os clientes ou grupos podem ser autorizados a executar consultas de objetos ou entidades nas grades de objetos.

Autorização do agente do DataGrid
: Clientes ou grupos podem ser autorizados a permitir que os agentes DataGrid sejam implementados a um ObjectGrid.

Autorização de Mapa do Lado do Servidor
: Clientes ou grupos podem ser autorizados a replicar um mapa do servidor no lado do cliente ou criar um índice dinâmico para o mapa do servidor.

Autorização de administração
: Clientes ou grupos podem ser autorizados a executar tarefas de administração.

Nota: Se você já tiver a segurança ativada para seu backend, lembre-se de que estas configurações de segurança não são mais suficientes para proteger seus dados. As configurações de segurança do seu banco de dados ou outro datastore não são, de forma alguma, transferidas para o seu cache. É necessário proteger separadamente os dados que agora são armazenados em cache utilizando o mecanismo de segurança do eXtreme Scale, incluindo segurança no nível de autenticação, autorização e transporte.

Segurança de Grade

A segurança de grade do WebSphere eXtreme Scale garante que um servidor que está se juntando tenha as credenciais certas, assim um servidor doloso não pode juntar-se à grade. Ela utiliza um mecanismo de cadeia de segredos compartilhados para este propósito.

Todos os servidores WebSphere eXtreme Scale, incluindo servidores de catálogo, concordam quanto a uma cadeia de segredo compartilhado. Quando um servidor se junta à grade, ele é desafiado a apresentar a cadeia secreta. Se a cadeia secreta do servidor que está se juntando corresponder à cadeia no servidor presidente ou servidor de catálogo, o servidor que está se juntando é aceito. Se a cadeia não corresponder, o pedido de junção é rejeitado.

Não é seguro enviar um segredo em texto não-criptografado. A infraestrutura de segurança do WebSphere eXtreme Scale fornece um plug-in do gerenciador de token seguro para permitir que o servidor proteja este segredo antes de enviar. Você deve decidir como implementar a operação de proteção. O WebSphere eXtreme Scale fornece uma implementação pronta para usar, na qual a operação segura é implementada para criptografar e assinar o segredo.

A cadeia do segredo é configurada no arquivo `server.properties`. Consulte o "Arquivo de Propriedades do Servidor" na página 189 para obter mais informações sobre a propriedade authenticationSecret.

Plug-in SecureTokenManager

Um plug-in do gerenciador de token de segurança é representado pela interface com.ibm.websphere.objectgrid.security.plugins.SecureTokenManager.

Para obter mais informações sobre o plug-in SecureTokenManager, consulte aDocumentação da API SecureTokenManager.

O método generateToken(Object) obtém um objeto, e depois gera um token que não pode ser compreendido pelos outros. O método verifyTokens(byte[]) faz o processo inverso: o método converte o token de volta ao objeto original.

Uma implementação SecureTokenManager simples usa um algoritmo de codificação simples, como um algoritmo exclusivo ou (XOR), para codificar o objeto na forma serializada e depois usa o algoritmo de codificação correspondente para codificar o token. Essa implementação não é segura.

O WebSphere eXtreme Scale fornece uma implementação imediatamente disponível para esta interface.

A implementação padrão utiliza um par de chaves para assinar e verificar a assinatura e utiliza uma chave secreta para criptografar o conteúdo. Cada servidor tem um armazenamento de chaves de tipo JCKES para armazenar o par de chaves, uma chave privada e uma chave pública e uma chave secreta. O armazenamento de chaves tem que ser do tipo JCKES para armazenar as chaves secretas.

Estas chaves são utilizadas para criptografar e assinar ou verificar a cadeia de segredo na extremidade de envio. Além disso, o token está associado ao tempo de expiração. Na extremidade de recebimento, os dados são verificados, decriptografados e comparados com a cadeia de segredo do receptor. Os protocolos de comunicação Secure Sockets Layer (SSL) não são necessários entre um par de

servidores para autenticação, porque as chaves privadas e as chaves públicas servem para a mesma finalidade. No entanto, se a comunicação do servidor não for criptografada, os dados poderão ser roubados por violação na comunicação. Como o token expira em breve, a ameaça de ataque à reprodução é minimizada. Esta possibilidade é significativamente reduzida se todos os servidores forem implementados atrás de um firewall.

A desvantagem desta abordagem é que os administradores do WebSphere eXtreme Scale precisam gerar chaves e transportá-las para todos os servidores, o que pode causar violação de segurança durante o transporte.

Configuração

Consulte Propriedades do servidor para obter mais informações sobre as propriedades que você usa para configurar o gerenciador de token seguro.

Ativando a Segurança Local

O WebSphere eXtreme Scale fornece vários terminais de segurança para integrar mecanismos customizados. No modelo de programação local, a principal função de segurança é a autorização e não possui suporte à autenticação. É necessário autenticar fora do WebSphere Application Server já existente. Entretanto, são fornecidos plug-ins para obter e validar objetos Subject.

As seguintes seções descrevem as duas maneiras nas quais é possível ativar a segurança local:

É possível utilizar o arquivo XML do ObjectGrid para definir um ObjectGrid e ativar a segurança para esse ObjectGrid. O arquivo `secure-objectgrid-definition.xml`, que é utilizado na amostra do aplicativo corporativo ObjectGridSample, é mostrado no seguinte exemplo. Configure o atributo securityEnabled como `true` para ativar a segurança.

```
<objectGrids>
    <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
        authorizationMechanism="AUTHORIZATION_MECHANISM_JAAS">
    ...
</objectGrids>
```

Também é possível ativar a segurança programaticamente. Para criar um ObjectGrid usando o método ObjectGrid.setSecurityEnabled, chame o seguinte método na interface ObjectGrid:

```
/**
 * Enable the ObjectGrid security
 */
void setSecurityEnabled();
```

Para obter informações adicionais, consulte na documentação da API.

Autenticação de Cliente do Aplicativo

A autenticação de cliente do aplicativo consiste em ativar a autenticação de credencial e de segurança cliente/servidor e de configurar um autenticador e um gerador de credencial de sistema.

Ativação da Segurança do Cliente-Servidor

A segurança deve ser ativada no cliente e no servidor para poder autenticar-se com êxito com o ObjectGrid.

Ativar a Segurança do Cliente

O WebSphere eXtreme Scale fornece um arquivo de amostra de propriedade do cliente, o arquivo `sampleClient.properties` no diretório *WAS_HOME*`/optionalLibraries/ObjectGrid/properties` para uma instalação do WebSphere Extended Deployment ou no diretório `/ObjectGrid/properties` em uma instalação do servidor combinada. É possível modificar este arquivo de gabarito com valores apropriados. Configure a propriedade securityEnabled no arquivo `objectgridClient.properties` para `true`. A propriedade securityEnabled indica se a segurança está ativada. Quando um cliente se conecta a um servidor, os valores no lado do cliente e do servidor devem ser ambos `true` ou ambos `false`. Por exemplo, se a segurança do servidor conectado estiver ativada, o valor da propriedade deverá ser configurado como `true` na lado do cliente para que o cliente se conecte ao servidor.

A interface com.ibm.websphere.objectgrid.security.config.ClientSecurityConfiguration representa o arquivo `security.ogclient.props`. É possível utilizar a API pública com.ibm.websphere.objectgrid.security.config.ClientSecurityConfigurationFactory para criar uma instância desta interface com valores padrão ou é possível criar uma instância passando o arquivo de propriedades de segurança do cliente do ObjectGrid. O arquivo `security.ogclient.props` contém outras propriedades. Consulte a Documentação da API do ClientSecurityConfiguration e a Documentação da API do ClientSecurityConfigurationFactory para obter mais detalhes.

Ativação da Segurança do Servidor

Para ativar a segurança no lado do servidor, é possível configurar a propriedade securityEnabled no arquivo `security.xml` para `true`. Use um arquivo XML do descritor de segurança para especificar a configuração de segurança da grade para isolar a configuração de segurança em toda a grade de uma configuração que não seja de segurança.

Ativando a Autenticação de Credencial

Após o cliente do eXtreme Scale recuperar o objeto Credential utilizando o objeto CredentialGenerator, o objeto Credential é enviado junto o pedido do cliente para o servidor eXtreme Scale. O servidor autentica o objeto de Credencial antes de processar a solicitação. Se o objeto de Credencial for autenticado com êxito, um objeto Subject será retornado para representar este objeto de Credencial. Este objeto Subject é então utilizado para autorizar o pedido.

Configure credentialAuthentication nos arquivos de propriedades do cliente e do servidor para ativar a autenticação de credencial. Para obter mais informações, consulte “Arquivo de Propriedades do Cliente” na página 195 e “Arquivo de Propriedades do Servidor” na página 189.

As tabelas a seguir exibem qual mecanismo de autenticação será utilizado em diferentes configurações.

Tabela 9. Autenticação de credencial nas configurações do cliente e do servidor

Autenticação de Credencial do Cliente	Autenticação de Credencial do Servidor	Result
Não	Nunca	Desativada

Tabela 9. Autenticação de credencial nas configurações do cliente e do servidor (continuação)

Autenticação de Credencial do Cliente	Autenticação de Credencial do Servidor	Result
Não	Suportado	Desativada
Não	Requerido	Error case
Suportado	Nunca	Desativada
Suportado	Suportado	Ativado
Suportado	Requerido	Ativado
Requerido	Nunca	Error case
Requerido	Suportado	Ativado
Requerido	Requerido	Ativado

Configuração de um autenticador

O servidor eXtreme Scale utiliza o plug-in do Autenticador para autenticar o objeto Credential. Uma implementação da interface Autenticador obtém o objeto Credential e, em seguida, autentica-o para um registro do usuário, como por exemplo, um servidor Lightweight Directory Access Protocol (LDAP), etc. O eXtreme Scale não fornece uma configuração de registro. A conexão com um registro do usuário e a autenticação nele devem ser implementadas neste plug-in.

Por exemplo, a implementação de um Autenticador extrai o ID do usuário e a senha da credencial, utiliza-os para conectar-se e validar em um servidor LDAP e cria um objeto Subject como resultado da autenticação. A implementação pode utilizar os módulos de login do Java Authentication and Authorization Service (JAAS). Um objeto Subject é retornado como resultado de autenticação.

É possível configurar o autenticador no arquivo XML descritor de segurança, como mostra o seguinte exemplo:

```
<?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>
```

Use a opção –clusterSecurityFile ao iniciar um servidor seguro para configurar o arquivo XML de segurança. Consulte o tutorial de segurança Java SE no *Visão Geral do Produto* para obter mais informações.

Configuração de um gerador de credencial do sistema

O gerador de credenciais do sistema é utilizado para representar um factory para a credencial do sistema. Uma credencial de sistema é semelhante para uma credencial de administrador. É possível certificar o elemento SystemCredentialGenerator no XML de segurança do catálogo, como mostrado no exemplo a seguir:

```
<systemCredentialGenerator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.UserPasswordCredentialGenerator">
         <property name="properties" type="java.lang.String" value="manager manager1" description="username password" />
    </systemCredentialGenerator>
```

Para demonstração, o nome de usuário e senha são armazenados em texto limpo. Não armazene o nome de usuário e senha em texto limpo em um ambiente de produção.

O WebSphere eXtreme Scale fornece um gerador de credencial de sistema padrão, que usa as credenciais do servidor. Se você não especificar explicitamente o gerador de credencial de sistema, este gerador de credencial de sistema padrão é usado.

Autorização do Aplicativo Cliente

A autorização do cliente do aplicativo consiste de classes de permissão do ObjectGrid, de mecanismos de autorização, de um período de verificação de permissão e de autorização de acesso apenas pelo criador.

Para o eXtreme Scale, a autorização se baseia no objeto Subject e nas permissões. O produto suporta dois tipos de mecanismos de autorização: Java Authentication and Authorization Service (JAAS) e autorização customizada.

Classes de permissão do ObjectGrid

A autorização é baseada em permissões. A seguir há quatro tipos diferentes de classes de permissão.

- A classe MapPermission representa as permissões para acessar os dados nos mapas do ObjectGrid.
- A classe ObjectGridPermission representa as permissões para acessar o ObjectGrid.
- A classe ServerMapPermission representa as permissões para acessar os mapas do ObjectGrid no lado do servidor a partir de um cliente.
- A classe AgentPermission representa as permissões para iniciar um agente no lado do servidor.

Para obter mais informações sobre as APIs e permissões associadas, consulte o tópico sobre a programação de autorização do cliente no *Guia de Programação*.

Período de verificação de permissão

O eXtreme Scale suporta armazenamento em cache dos resultados da verificação de permissão de mapa por motivo de desempenho. Sem este mecanismo, quando um método listado na Lista de Métodos com suas permissões necessárias é chamado, o tempo de execução chama o mecanismo de autorização configurado para autorizar acesso. Com este período de verificação de permissão configurado, o mecanismo de autorização é chamado periodicamente com base no período de verificação de permissão.

As informações de autorização da permissão são baseadas no objeto Subject. Quando um cliente tenta acessar os métodos, o tempo de execução do eXtreme Scale consulta o cache com base no objeto Subject. Se o objeto não puder ser localizado no cache, o tempo de execução verifica as permissões concedidas para este objeto Subject, e, então, armazena as permissões em um cache.

O período de verificação de permissão deve ser definido antes da inicialização do ObjectGrid. O período de verificação de permissão pode ser configurado de duas maneiras:

É possível utilizar o arquivo XML do ObjectGrid para definir um ObjectGrid e configurar o período de verificação de permissão. No exemplo a seguir, o período de verificação de permissão é configurado para 45 segundos:

```
<objectGrids>
 <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
 authorizationMechanism="AUTHORIZATION_MECHANISM_JAAS"
 permissionCheckPeriod="45">
  <bean id="bean id="TransactionCallback"
className="com.ibm.websphere.samples.objectgrid.HeapTransactionCallback" />
...
</objectGrids>
```

Se desejar criar um ObjectGrid com APIs, chame o seguinte método para configurar o período de verificação de permissão. Este método pode ser chamado apenas antes da inicialização da instância do ObjectGrid. Este método se aplica apenas ao modelo de programação do eXtreme Scale local quando você utiliza a instância do ObjectGrid diretamente.

```
/**
 * This method takes a single parameter indicating how often you
 * want to check the permission used to allow a client access. If the
 * parameter is 0 then every single get/put/update/remove/evict call
 * asks the authorization mechanism, either JAAS authorization or custom
 * authorization, to check if the current subject has permission. This might be
 * prohibitively expensive from a performance point of view depending on
 * the authorization implementation, but if you need to have ever call check the
 * authorization mechanism, then set the parameter to 0.
 * Alternatively, if the parameter is > 0 then it indicates the number
 * of seconds to cache a set of permissions before returning to
 * the authorization mechanism to refresh them. This value provides much
 * better performance, but if the back-end
 * permissions are changed during this time then the ObjectGrid can
 * allow or prevent access even though the back-end security
 * provider was modified.
 *
 * @param period the permission check period in seconds.
 */
void setPermissionCheckPeriod(int period);
```

Autorização de acesso apenas pelo criador

A autorização de acesso apenas pelo criador garante que apenas o usuário (representado pelos objetos Principal associados a ele) que insere a entidade no mapa ObjectGrid possa acessar (ler, atualizar, invalidar e remover tal entrada.

O modelo existente de autorização do mapa do ObjectGrid é baseado no tipo de acesso mas não em entradas de dados. Em outras palavras, um usuário possui um tipo de acesso específico, tal como read, write, insert, delete ou invalidate, para todo os dados no mapa ou em nenhum dos dados. Entretanto, o eXtreme Scale não autoriza usuários para a entrada de dados individual. Este recurso oferece uma nova maneira de autorizar usuários para entradas de dados.

Em um cenário onde usuários diferentes acessam diferentes conjuntos de dados, este modelo pode ser útil. Quando o usuário carrega dados do armazenamento persistente nos mapas do ObjectGrid, o acesso pode ser autorizado pelo armazenamento persistente. Neste caso, não há necessidade de outra autorização na camada do mapa do ObjectGrid. É necessário apenas garantir que a pessoa que carrega os dados no mapa possa acessá-lo, permitindo o recurso de acesso apenas pelo criador.

Há três diferentes modos de acesso apenas pelo criador:

desativada
O recurso de acesso apenas pelo criador é desativado.

complement
O recurso de acesso apenas pelo criador é ativado para complementar a autorização do mapa. Em outras palavras, a autorização de mapa e o recurso de acesso apenas pelo criador serão efetivados. Portanto, é possível limitar ainda mais as operações nos dados. Por exemplo, o criador não pode invalidar os dados.

supersede
O recurso de acesso apenas pelo criador é ativado para substituir a autorização do mapa. Em outras palavras, o recurso de acesso apenas pelo criador substituirá a autorização do mapa e nenhuma autorização de mapa será feita.

É possível configurar o modo de acesso apenas pelo criador de duas maneiras:

É possível utilizar o arquivo XML do ObjectGrid para definir um ObjectGrid e configurar o acesso de modo apenas pelo criador para `disabled`, `complement` ou `supersede`, como mostra o seguinte exemplo:

```
<objectGrids>
    <objectGrid name="secureClusterObjectGrid" securityEnabled="true"
         accessByCreatorOnlyMode="supersede"
        <bean id="TransactionCallback"
              classname="com.ibm.websphere.samples.objectgrid.HeapTransactionCallback" />
    ...
</objectGrids>
```

Se desejar criar um ObjectGrid programaticamente, é possível chamar o método a seguir para configurar o modo de acesso apenas pelo criador. A chamada deste método aplica-se somente ao modelo de programação local do eXtreme Scale quando você instancia diretamente a instância do ObjectGrid:

```
/**
 * Set the "access by creator only" mode.
 * Enabling "access by creator only" mode ensures that only the user (represented
 * by the Principals associated with it), who inserts the record into the map,
 * can access (read, update, invalidate, and remove) the record.
 * The "access by creator only" mode can be disabled, or can complement the
 * ObjectGrid authorization model, or it can supersede the ObjectGrid
 * authorization model. The default value is disabled:
 * {@link SecurityConstants#ACCESS_BY_CREATOR_ONLY_DISABLED}.
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_DISABLED
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_COMPLEMENT
 * @see SecurityConstants#ACCESS_BY_CREATOR_ONLY_SUPERSEDE
 *
 * @param accessByCreatorOnlyMode the access by creator mode.
 *
 * @since WAS XD 6.1 FIX3
*/
void setAccessByCreatorOnlyMode(int accessByCreatorOnlyMode);
```

Para ilustrar ainda mais, considere um cenário no qual uma conta de mapa do ObjectGrid está em uma grade financeira e Manager1 e o Employee1 são os dois usuários. A política de autorização do eXtreme Scale concede todas as permissões de acesso para o Manager1 e apenas a permissão de acesso de leitura para o Employee1. A política do JAAS para a autorização do mapa do ObjectGrid é mostrada no seguinte exemplo:

```
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
    Principal com.acme.PrincipalImpl "Manager1" {
    permission com.ibm.websphere.objectgrid.security.MapPermission
        "banking.account", "all"
};
grant codebase "http://www.ibm.com/com/ibm/ws/objectgrid/security/PrivilegedAction"
    Principal com.acme.PrincipalImpl "Employee1" {
    permission com.ibm.websphere.objectgrid.security.MapPermission
        "banking.account", "read"
};
```

Considere como o recurso de acesso apenas pelo criador afeta a autorização:

- **disabled** Se o recurso de acesso apenas pelo criador estiver desativado, a autorização de mapa não é diferente. O usuário ″Manager1″ pode acessar todos os dados no mapa ″account″. O usuário ″Employee1″ pode ler todos os dados no mapa mas não pode atualizar, invalidar, remover quaisquer dados no mapa.
- **complement** Se o recurso de acesso apenas pelo criador estiver ativado com a opção ″complement″, a autorização de mapa e a autorização de acesso apenas pelo criador estarão em vigor. O usuário ″Manager1″ pode acessar os dados no mapa ″account″, mas apenas se o usuário sozinho os carregou no mapa. O usuário ″Employee1″ pode ler os dados no mapa ″account″, mas apenas se tal usuário sozinho os carregou no mapa. (Entretanto, este usuário não pode atualizar, invalidar ou remover dados no mapa).
- **supersede** Se o recurso de acesso apenas pelo criador estiver ativado com a opção ″supersede″, a autorização do mapa não será imposta. A autorização de acesso apenas pelo criador será a política apenas de autorização. O usuário ″Manager1″ possui o mesmo privilégio que no modo ″complement″: este usuário pode acessar os dados no mapa ″account″ apenas se o mesmo usuário carregou os dados no mapa. Entretanto, o usuário ″Employee1″ agora possui acesso total aos dados no mapa ″account″ se este usuários os carregou no mapa. Em outras palavras, a política de autorização definida na política do Java Authentication and Authorization Service (JAAS) não será imposta.

Transport Layer Security e Secure Sockets Layer

O WebSphere eXtreme Scale suporta tanto TCP/IP quanto TLS/SSL (TCP/IP and Transport Layer Security/Secure Sockets Layer) para comunicação segura entre cliente e servidor em modelos de implementação dinâmica.

O TLS/SSL fornece comunicação segura entre o cliente e o servidor. O mecanismo de comunicação que é usado depende do valor do parâmetro transportType que está especificado nos arquivos de configuração do cliente e servidor.

Você pode configurar a propriedade transportType nos seguintes arquivos de configuração de cliente e servidor:

- Para configurar a propriedade na configuração de segurança do cliente, consulte “Arquivo de Propriedades do Cliente” na página 195.
- Para configurar a propriedade na configuração de segurança do servidor de contêineres, consulte “Arquivo de Propriedades do Servidor” na página 189.
- Para configurar a propriedade na configuração de segurança do servidor de catálogos, consulte “Arquivo de Propriedades do Servidor” na página 189.

Tabela 10. Protocolo de Transporte a Ser Utilizado nas Configurações de Transporte do Cliente e Transporte do Servidor

Propriedade transportType do cliente	Propriedade transportType do servidor	Protocolo resultante
TCP/IP	TCP/IP	TCP/IP
TCP/IP	Suportado pelo SSL	TCP/IP
TCP/IP	Requerido pelo SSL	Erro
Suportado pelo SSL	TCP/IP	TCP/IP
Suportado pelo SSL	Suportado pelo SSL	SSL (se o SSL falhar, TCP/IP)
Suportado pelo SSL	Requerido pelo SSL	SSL
Requerido pelo SSL	TCP/IP	Erro
Requerido pelo SSL	Suportado pelo SSL	SSL
Requerido pelo SSL	Requerido pelo SSL	SSL

Quando é usado SSL, os parâmetros de configuração SSL devem ser fornecidos tanto do lado do cliente quanto do servidor. Em um ambiente Java SE, a configuração SSL é configurada nos arquivos de propriedades do cliente ou servidor. Se o cliente ou servidor estiver em um WebSphere Application Server, então você pode usar o transports suporte de segurança do WebSphere Application Server para configurar os parâmetros SSL.

Configuração do arquivo `orb.properties` para Suporte de Segurança de Transporte

Use o TLS/SSL quando a propriedade transportType possuir um valor ″SSL-Suportado″.

Para suportar o transporte seguro em um ambiente Java Platform, Standard Edition, é necessário modificar o arquivo “Arquivo de Propriedades ORB” na página 197 para incluir as seguintes propriedades:

```
# IBM JDK properties
org.omg.CORBA.ORBClass=com.ibm.CORBA.iiop.ORB
org.omg.CORBA.ORBSingletonClass=com.ibm.rmi.corba.ORBSingleton
javax.rmi.CORBA.StubClass=com.ibm.rmi.javax.rmi.CORBA.StubDelegateImpl
javax.rmi.CORBA.PortableRemoteObjectClass=com.ibm.rmi.javax.rmi.PortableRemoteObject
javax.rmi.CORBA.UtilClass=com.ibm.ws.orb.WSUtilDelegateImpl

# WS Plugins
com.ibm.CORBA.ORBPluginClass.com.ibm.ws.orbimpl.transport.WSTransport
com.ibm.CORBA.ORBPluginClass.com.ibm.ws.orbimpl.WSORBPropertyManager
com.ibm.CORBA.ORBPluginClass.com.ibm.ISecurityUtilityImpl.SecurityPropertyManager

# WS Interceptors
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ws.objectgrid.corba.ObjectGridInitializer
org.omg.PortableInterceptor.ORBInitializerClass.com.ibm.ISecurityLocalObjectBaseL13Impl.SecurityComponentFactory

# WS ORB & Plugins properties
com.ibm.ws.orb.transport.ConnectionInterceptorName=com.ibm.ISecurityLocalObjectBaseL13Impl.SecurityConnectionInterceptor
com.ibm.ws.orb.transport.WSSSLClientSocketFactoryName=com.ibm.ws.security.orbssl.WSSSLClientSocketFactoryImpl

com.ibm.CORBA.TransportMode=Pluggable
com.ibm.CORBA.ServerName=ogserver
```

Configuração de Parâmetros SSL para Clientes do eXtreme Scale

É possível configurar os parâmetros SSL para os clientes da seguinte forma:

1. Crie um objeto com.ibm.websphere.objectgrid.security.config.SSLConfiguration usando o com.ibm.websphere.objectgrid.security.config. Classe de factory ClientSecurityConfigurationFactory. Para obter mais detalhes, consulte a Documentação da API do ClientSecurityConfigurationFactory.
2. Configure os parâmetros no arquivo `client.propeties` e, em seguida, use o método ClientSecurityConfigurationFactory.getClientSecurityConfiguration(String) para preencher a instância do objeto.

Consulte a seção nas propriedades do cliente de segurança no “Arquivo de Propriedades do Cliente” na página 195 para obter exemplos de propriedades que podem ser configuradas em um cliente.

Configuração de Parâmetros SSL para Servidores do eXtreme Scale

Os parâmetros SSL são configurados para servidores usando um arquivo de propriedades do servidor, como por exemplo o arquivo `server.properties` referido acima. Este arquivo de propriedades pode ser transmitido como um parâmetro ao iniciar um servidor do eXtreme Scale. Para obter informações adicionais sobre os parâmetros SSL, você pode configurar servidores do eXtreme Scale, consulte

"Arquivo de Propriedades do Servidor" na página 189.

Suporte a Segurança de Transporte no WebSphere Application Server

Quando um cliente do eXtreme Scale, o servidor de contêineres ou servidor de catálogos está em execução em um processo do WebSphere Application Server, a segurança de transporte do eXtreme Scale é gerenciada pelas configurações de transporte CSIV2 do Gerenciador de Aplicativos. Você não deve usar as propriedades do cliente e do servidor do eXtreme Scale para configurar as configurações SSL. Todas as configurações SSL devem ser especificadas na configuração do WebSphere Application Server.

Nota:

Se você definir as configurações do SSL com um arquivo de propriedades, as configurações existentes serão substituídas.

Autenticação de Grade

O plug-in do gerenciador de tokens seguros é outro plug-in usado para o servidor para efetuar autenticação. O plug-in do gerenciador de tokens seguros é representado pela interface com.ibm.websphere.objectgrid.security.plugins.SecureTokenManager.

O método generateToken(Object) obtém uma proteção de objeto, e depois gera um token que não pode ser compreendido pelos outros. O método verifyTokens(byte[]) faz o processo inverso: converte o token de volta ao objeto original.

Uma implementação SecureTokenManager simples usa um algoritmo de codificação simples, como um algoritmo XOR, para codificar o objeto na forma serializada e depois usa o algoritmo de codificação correspondente para decodificar o token. Esta implementação não é segura e é fácil de ser interrompida.

Implementação padrão do **WebSphere eXtreme Scale**

O WebSphere eXtreme Scale fornece uma implementação imediatamente disponível para esta interface. Esta implementação padrão utiliza um par de chaves para assinar e verificar a assinatura e utiliza uma chave secreta para criptografar o conteúdo. Cada servidor tem um armazenamento de chaves de tipo JCKES para armazenar o par de chaves, uma chave privada e uma chave pública e uma chave secreta. O armazenamento de chaves tem que ser do tipo JCKES para armazenar as chaves secretas. Estas chaves são utilizadas para criptografar e assinar ou verificar a cadeia de segredo na extremidade de envio. Além disso, o token está associado ao tempo de expiração. Na extremidade de recebimento, os dados são verificados, decriptografados e comparados com a cadeia de segredo do receptor. Os protocolos de comunicação Secure Sockets Layer (SSL) não são necessários entre um par de servidores para autenticação, porque as chaves privadas e as chaves públicas servem para a mesma finalidade. No entanto, se a comunicação do servidor não for criptografada, os dados poderão ser roubados por violação na comunicação. Como o token expira em breve, a ameaça de ataque à reprodução é minimizada. Esta possibilidade é significativamente reduzida se todos os servidores forem implementados atrás de um firewall.

A desvantagem desta abordagem é que os administradores do WebSphere eXtreme Scale precisam gerar chaves e transportá-las para todos os servidores, o que pode causar violação de segurança durante o transporte.

Segurança do Java Management Extensions (JMX)

É possível proteger as chamadas de beans gerenciados (MBean) em um ambiente de implementação dinâmica.

Para obter mais informações sobre MBeans disponíveis, consulte "Utilizando beans gerenciados (MBeans) para administrar seu ambiente" na página 265.

Na topologia de implementação dinâmica, os MBeans são diretamente hospedados nos servidores de catálogos e servidores de contêineres. Em geral, a segurança JMX em uma topologia de implementação dinâmica segue a especificação de segurança JMX conforme definido na Especificação JavaTM Management Extensions (JMX). Ela consiste nas três partes a seguir:

1. Autenticação - O cliente remoto precisa ser autenticado no servidor do conector.
2. Controle de acesso - O controle de acesso de MBean limita quem pode acessar as informações de MBean e quem pode executar as operações de MBean.
3. Transporte seguro - O transporte entre o cliente JMX e o servidor pode ser protegido utilizando TLS/SSL.

Autenticação

O JMX fornece métodos para os servidores conectores para autenticar os clientes remotos. Para o conector RMI, a autenticação é concluída fornecendo um objeto que implementa a interface JMXAuthenticator quando o servidor conector é criado. Assim, o eXtreme Scale implementa esta interface JMXAuthenticator para utilizar o plug-in do ObjectGrid Authenticator para autenticar os clientes remotos. Consulte o tutorial de segurança no *Visão Geral do Produto* para obter detalhes sobre como o eXtreme Scale autentica um cliente.

O cliente JMX segue as APIs do JMX para fornecer credenciais para conectar-se com o servidor conector. A estrutura JMX passa a credencial para o servidor conector e, então, chama a implementação do JMXAuthenticator para autenticação. Conforme descrito anteriormente, a implementação do JMXAuthenticator então delega a autenticação para a implementação do Autenticador doObjectGrid.

Revise o exemplo a seguir que descreve como conectar-se a um servidor conector com uma credencial:

```
javax.management.remote.JMXServiceURL jmxUrl = new JMXServiceURL(
        "service:jmx:rmi:///jndi/rmi://localhost:1099/objectgrid/MBeanServer");

    environment.put(JMXConnector.CREDENTIALS, new UserPasswordCredential("admin", "xxxxxx"));

    // Create the JMXCconnectorServer
    JMXConnector cntor = JMXConnectorFactory.newJMXConnector(jmxUrl, null);

    // Connect and invoke an operation on the remote MBeanServer
    cntor.connect(environment);
```

No exemplo anterior, um UserPasswordCredential é fornecido com o ID do usuário configurado como admin e a senha configurada como xxxxx. Este objeto UserPasswordCredential é configurado no mapa do ambiente, que é utilizado no método JMXConnector.connect(Map). Este objeto UserPasswordCredential é então passado para o servidor pela estrutura JMX, e finalmente passado para a estrutura de autenticação do ObjectGrid para a autenticação.

O modelo de programação do cliente segue estritamente a especificação JMX.

Controle de Acesso

Um servidor MBean JMX pode ter acesso a informações sensitivas e pode executar operações sensitivas. O JMX fornece o controle de acesso necessário que identifica quais clientes podem acessar tais informações e quem pode executar tais operações. o controle de acesso é integrado no modelo de segurança Java padrão por meio da definição de permissões que controlam o acesso ao servidor MBean e às suas operações.

Para controle de acesso de operação ou autorização do JMX, oeXtreme Scale conta com o suporte de JAAS fornecido pela implementação JMX. Em qualquer ponto determinado na execução de um programa, há um conjunto atual de permissões que um encadeamento de execuções contém. Quando um destes encadeamentos chama uma operação de especificação JMX, estes são conhecidos como as permissões contida. Quando uma operação JMX é executada, uma verificação de segurança é feita para verificar se a permissão necessária é incluída pela permissão contida.

A definição de política do MBean segue o formato da política Java. Por exemplo, a seguinte política concede a todos os assinantes e todas as bases de código o direito de recuperar o endereço JMX do servidor para o PlacementServiceMBean, mas com restrição ao domínio com.ibm.websphere.objectgrid.

```
grant {
    permission javax.management.MBeanPermission
        "com.ibm.websphere.objectgrid.management.PlacementServiceMBean#retrieveServerJMXAddress
        [com.ibm.websphere.objectgrid:*,type=PlacementService]",
        "invoke";
}
```

O exemplo de política a seguir pode ser utilizado para concluir a autorização baseada na identidade do cliente remoto. A política concede a mesma permissão MBean, conforme mostrado no exemplo anterior, exceto apenas para usuários com o nome X500Principal como CN=Administrator,OU=software,O=IBM,L=Rochester,ST=MN,C=US.

```
grant principal javax.security.auth.x500.X500Principal "CN=Administrator,OU=software,O=IBM,L=Rochester,ST=MN,C=US" {
    permission javax.management.MBeanPermission
        "com.ibm.websphere.objectgrid.management.PlacementServiceMBean#retrieveServerJMXAddress
        [com.ibm.websphere.objectgrid:*,type=PlacementService]",
        "invoke";
}
```

As políticas do Java são verificadas somente se o gerenciador de segurança estiver ativado. Inicie os servidores de catálogos e servidores de contêineres com o argumento JVM -Djava.security.manager para forçar o controle de acesso da operação MBean.

Transporte Seguro

O transporte entre o cliente e o servidor JMX pode ser protegido utilizando TLS/SSL. Se o transportType do servidor de catálogos ou servidor de contêineres for configurado como SSL_Required ou SSL_Supported, então, é necessário utilizar o SSL para conectar-se ao servidor JMX.

Para utilizar SSL, é necessário configurar o trust store, tipo de trust store e a senha trust store no cliente MBean utilizando propriedades do sistema -D:

1. -Djavax.net.ssl.trustStore=TRUST_STORE_LOCATION
2. -Djavax.net.ssl.trustStorePassword=TRUST_STORE_PASSWORD

3. -Djavax.net.ssl.trustStoreType=TRUST_STORE_TYPE

Se você utilizar com.ibm.websphere.ssl.protocol.SSLSocketFactory como seu socket factory SSL em seu arquivo JAVA_HOME/jre/lib/security/java.security, então, utilize as seguintes propriedades:

1. -Dcom.ibm.ssl.trustStore=TRUST_STORE_LOCATION
2. -Dcom.ibm.ssl.trustStorePassword=TRUST_STORE_PASSWORD
3. -Dcom.ibm.ssl.trustStoreType=TRUST_STORE_TYPE

Arquivo XML Descritor de Segurança

Use um arquivo descritor de segurança do ObjectGrid para configurar uma topologia de implementação do eXtreme Scale com segurança ativada. Amostras de arquivos XML são fornecidas neste tópico para ilustrar várias configurações.

Cada elemento e atributo do arquivo XML do cluster está descrito na lista a seguir. Utilize os exemplos para aprender como utilizar esses elementos e atributos para configurar o ambiente.

Elemento securityConfig

O elemento securityConfig é o elemento de nível superior do arquivo XML de segurança do ObjectGrid. Este elemento configura o espaço de nomes do arquivo e o local do esquema. O esquema está definido no arquivo `objectGridSecurity.xsd`.

- Número de ocorrências: Uma
- Elementos filho : segurança

Elemento security

Utilize o elemento security para definir uma segurança do ObjectGrid.

- Número de ocorrências: Uma
- Elementos-filho: authenticator, adminAuthorization e systemCredentialGenerator

Atributos

securityEnabled
Ativa a segurança para a grade quando configurado para true. O valor padrão é false Se o valor for configurado como false, a segurança no escopo da grade será desativada. Para obter informações adicionais, consulte “Segurança de Grade” na página 286. (Opcional)

singleSignOnEnabled
Permite que o cliente se conecte a qualquer servidor depois que ele for autenticado com um dos servidores, se o valor for configurado para true. Caso contrário, um cliente deverá se autenticar com cada servidor antes de poder se conectar.O valor padrão é false. (Opcional)

loginSessionExpirationTime
Especifica a quantidade de tempo, em segundos, antes de a sessão de login expirar. Se a sessão de login expirar, o cliente precisa autenticar-se novamente. (Opcional)

adminAuthorizationEnabled
Ativa a autorização administrativa. Se o valor for configurado para true, todas as tarefas administrativas precisarão de autorização. O mecanismo de

autorização utilizado é especificado pelo valor do atributo adminAuthorizationMechanism. O valor padrão é false. (Opcional)

adminAuthorizationMechanism
Indica qual mecanismo de autorização deve ser usado. O WebSphere eXtreme Scale suporta dois mecanismos de autorização, JAAS (Java Authentication and Authorization Service) e autorização personalizada. O mecanismo de autorização JAAS utiliza a abordagem baseada em política JAAS padrão. Para especificar JAAS como o mecanismo de autorização, configure o valor como AUTHORIZATION_MECHANISM_JAAS. O mecanismo de autorização customizado utiliza uma implementação de AdminAuthorization conectada pelo usuário. Para especificar um mecanismo de autorização customizado, configure o valor como AUTHORIZATION_MECHANISM_CUSTOM. Para obter informações adicionais sobre como estes dois mecanismos são utilizados, consulte "Autorização do Aplicativo Cliente" na página 290. (Opcional)

O arquivo `security.xml` a seguir é uma configurações de amostra para ativar a segurança da grade do eXtreme Scale.

security.xml

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

    <security securityEnabled="true" singleSignOnEnabled="true"
        loginSessionExpirationTime="20"
        adminAuthorizationEnabled="true"
        adminAuthorizationMechanism="AUTHORIZATION_MECHANISM_JAAS" >

        <authenticator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator">
        </authenticator>

        <systemCredentialGenerator className ="com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator">
            <property name="properties" type="java.lang.String" value="runAs" description="Using runAs subject" />
        </systemCredentialGenerator>

    </security>
</securityConfig>
```

Elemento authenticator

Autentica clientes para servidores eXtreme Scale na grade. A classe especificada pelo atributo className deve implementar a interface com.ibm.websphere.objectgrid.security.plugins.Authenticator. O autenticador pode utilizar propriedades para chamar métodos na classe que é especificada pelo atributo className. Consulte o elemento de propriedade para obter mais informações sobre como utilizar as propriedades.

No exemplo do arquivo `security.xml` anterior, a classe com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator é especificada como o autenticador. Esta classe implementa a interface com.ibm.websphere.objectgrid.security.plugins.Authenticator.

- Número de ocorrências: zero ou uma
- Elemento filho: propriedade

Atributos

className
Especifica a classe que implementa a interface com.ibm.websphere.objectgrid.security.plugins.Authenticator. Use esta classe para autenticar clientes para os servidores na grade do eXtreme Scale. (Obrigatório)

Elemento adminAuthorization

Use o elemento adminAuthorization para configurar o acesso administrativo para a grade.

- Número de ocorrências: zero ou uma
- Elemento filho: propriedade

Atributos

className
Especifica a classe que implementa a interface com.ibm.websphere.objectgrid.security.plugins.AdminAuthorization. (Obrigatório)

Elemento systemCredentialGenerator

Utilize um elemento systemCredentialGenerator para configurar um gerador de credenciais do sistema. Este elemento é aplicável apenas a um ambiente dinâmico. No modelo de configuração dinâmica, o servidor de contêineres dinâmicos se conecta ao servidor de catálogos como um cliente do eXtreme Scale e o servidor de catálogos pode se conectar ao servidor de contêineres do eXtreme Scale como um cliente também. O gerador de credenciais do sistema é utilizado para representar um depósito de informações para a credencial do sistema.

- Número de ocorrências: zero ou uma
- Elemento filho: propriedade

Atributos

className
Especifica a classe que implementa a interface com.ibm.websphere.objectgrid.security.plugins.CredentialGenerator. (Obrigatório)

Consulte o arquivo `security.xml` anterior para saber como utilizar um systemCredentialGenerator. Neste exemplo, o gerador de credenciais do sistema é um com.ibm.websphere.objectgrid.security.plugins.builtins. WSTokenCredentialGenerator, que recupera o objeto RunAs Subject a partir do encadeamento.

Elemento property

Chama o método set nas classes authenticator e adminAuthorization. O nome da propriedade corresponde a um método configurado no atributo className do elemento authenticator ou adminAuthorization.

- Número de ocorrências: zero ou mais
- Elemento filho: propriedade

Atributos

name
Especifica o nome da propriedade. O valor atribuído a esse atributo deve corresponder a um método set na classe que é fornecida como atributo className no bean que contém o atributo. Por exemplo, se o atributo className do bean for configurado como com.ibm.MyPlugin e o nome da propriedade que é fornecido for size, então, a classe com.ibm.MyPlugin deve ter um método setSize. (Obrigatório)

type
Especifica o tipo da propriedade. O tipo do parâmetro é transmitido ao método set identificado pelo atributo name. Os valores válidos são as primitivas Java, suas partes java.lang e java.lang.String. Os atributos name e type devem corresponder a uma assinatura de método no atributo className do bean. Por exemplo, se o nome for size e o tipo for int, então, deve existir um método setSize(int) na classe que é especificada como o atributo className para o bean. (Obrigatório)

value
Especifica o valor da propriedade. Este valor é convertido no tipo especificado pelo atributo type e, em seguida, é utilizado como um parâmetro na chamada para o método set identificado pelos atributos name e type. O valor deste atributo não é validado de nenhuma maneira. O implementador do plug-in deve verificar se o valor transmitido é válido. (Obrigatório)

description
Fornece uma descrição da propriedade (Opcional)

Consulte "Arquivo `objectGridSecurity.xsd`" na página 187 para obter mais informações.

Integração de Segurança com o WebSphere Application Server

O WebSphere eXtreme Scale fornece diversos recursos de segurança para integrar com a infraestrutura de segurança do WebSphere Application Server.

Integração de Autenticação

Quando os clientes e servidores do eXtreme Scale estão em execução no WebSphere Application Server e no mesmo domínio de segurança, é possível usar a infraestrutura de segurança do WebSphere Application Server para propagar as credenciais de autenticação do cliente para o servidor do eXtreme Scale. Por exemplo, se um servlet atuar como um cliente do eXtreme Scale para se conectar a um servidor do eXtreme Scale no mesmo domínio de segurança, e o servlet já estiver autenticado, é possível propagar o token de autenticação do cliente (servlet) para o servidor e, em seguida, usar a infraestrutura de segurança do WebSphere Application Server para converter o token de autenticação de volta para as credenciais do cliente.

Integração de segurança distribuída com o WebSphere Application Server

Para o modelo de ObjectGrid distribuído, a integração de segurança pode ser feita utilizando as seguintes classes:

`com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredentialGenerator`

`com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenAuthenticator`

`com.ibm.websphere.objectgrid.security.plugins.builtins.WSTokenCredential`

Consulte o "Autenticação de Cliente do Aplicativo" na página 287 para obter informações adicionais. O exemplo a seguir ilustra como utilizar a classe WSTokenCredentialGenerator:

```
/**
 * connect to the ObjectGrid Server.
 */
protected ClientClusterContext connect() throws ConnectException {
  ClientSecurityConfiguration csConfig = ClientSecurityConfigurationFactory
  .getClientSecurityConfiguration(proFile);

  CredentialGenerator gen = getWSCredGen();
```

```
  csConfig.setCredentialGenerator(gen);

  return objectGridManager.connect(csConfig, null);
}

/**
 * Get a WSTokenCredentialGenerator
 *
private CredentialGenerator getWSCredGen() {
    WSTokenCredentialGenerator gen = new WSTokenCredentialGenerator(
        WSTokenCredentialGenerator.RUN_AS_SUBJECT);
    return gen;
}
```

No lado do servidor, o WSTokenAuthentication pode ser utilizado como o autenticador para autenticar o objeto WSTokenCredential.

Integração de segurança local com o WebSphere Application Server

Para o modelo de ObjectGrid local, a integração de segurança pode ser feita utilizando as duas classes a seguir:

- com.ibm.websphere.objectgrid.security.plugins.builtins.WSSubjectSourceImpl
- com.ibm.websphere.objectgrid.security.plugins.builtins.WSSubjectValidationImpl

Para obter informações adicionais sobre estas classes, consulte as informações sobre a segurança local no *Guia de Programação*. É possível configurar a classe WSSubjectSourceImpl como o plug-in SubjectSource e a classe WSSubjectValidationImpl como o plug-in SubjectValidation.

Início e Encerramento de Servidores Seguros do eXtreme Scale

Os servidores frequentemente precisam ser seguros para seu ambiente de implementação, o que exige uma configuração específica.

Inicialização de um Servidor Seguro em um Ambiente Java SE

Você pode iniciar um serviço de catálogo ou servidores de contêineres da forma a seguir.

Iniciando um serviço de catálogo seguro do eXtreme Scale

A inicialização de um processo do serviço de catálogo seguro do eXtreme Scale exige mais dois arquivos de configuração de segurança:

Arquivo XML descritor de segurança: O arquivo XML descritor de segurança descreve as propriedades de segurança comuns para todos os servidores (incluindo servidores de catálogo e servidores de contêiner). Um exemplo de propriedade é a configuração do autenticador que representa o registro do usuário e o mecanismo de autenticação.

Arquivo de propriedades do servidor. O arquivo de propriedades do servidor configura as propriedades específicas de segurança para o servidor.

Quando você usa o comando startOgServer.sh ou startOgServer.cat para iniciar um processo seguro do serviço de catálogo do eXtreme Scale, você pode usar o -clusterSecurityFile ou -clusterSecurityUrl para configurar seguramente o arquivo XML do descritor de segurança como um tipo file ou URL, e pode usar –serverProps para configurar o arquivo de propriedades do servidor.

Início de um servidor de contêineres seguros do eXtreme Scale

A inicialização de um servidor de contêineres seguro do eXtreme Scale exige um arquivo de configuração de segurança:

- **Arquivo de propriedades do servidor:** O arquivo de propriedades do servidor configura as propriedades de segurança específicas para o servidor. Consulte o "Arquivo de Propriedades do Servidor" na página 189 para obter mais detalhes.

Quando você usa o comando startOgServer.sh ou startOgServer.cat para iniciar um servidor de contêineres seguro do eXtreme Scale, você pode usar –serverProps para configurar o arquivo de propriedades do servidor. Há mais formas para configurar o arquivo de propriedades de servidor, consulte o arquivo de propriedades do servidor para obter mais detalhes.

Para obter mais detalhes sobre como usar o comando startOgServer.sh ou startOgServer.bat e suas opções, consulte "Script startOgServer" na página 223.

Encerramento de um Servidor Seguro do eXtreme Scale

O encerramento de um processo seguro do serviço de catálogo do eXtreme Scale ou servidor de contêineres exige um arquivo de configuração de segurança:

- **Arquivo de propriedades do cliente:** O arquivo de propriedades do cliente pode ser usado para configurar as propriedades de segurança do cliente. As propriedades de segurança do cliente são necessárias para um cliente se conectar a um servidor seguro. Consulte o "Arquivo de Propriedades do Cliente" na página 195 para obter mais detalhes.

Quando você usa o comando stopOgServer.sh ou stopOgServer.cat para encerrar um processo seguro do serviço de catálogo do eXtreme Scale ou servidor de contêineres, você pode usar -clientSecurityFile para configurar as propriedades de segurança do cliente.

Para obter mais detalhes sobre como usar o comando stopOgServer.sh ou stopOgServer.cat e suas opções, consulte "Script stopOgServer" na página 229.

Iniciando um Servidor Seguro no WebSphere Application Server

A inicialização de um servidor seguro ObjectGrid em WebSphere Application Server é semelhante à inicialização de um servidor não seguro ObjectGrids, exceto que é necessário passar os arquivos de configuração de segurança. Em vez de usar o -[PROPERTY_FILE] (por exemplo -serverProps) no comando como no ambiente Java SE, use -D[PROPERTY_FILE] nos argumentos genéricos da Java Virtual Machine (JVM).

Iniciando um serviço de catálogo seguro no Websphere Application Server

Um servidor de catálogos contém dois níveis diferente de informações de segurança:

- `-Dobjectgrid.cluster.security.xml.url`: Especifica o arquivo objectGridSecurity.xml que descreve as propriedades de segurança comuns a todos os servidores (incluindo servidores de catálogos e servidores de contêineres). Um exemplo é a configuração do autenticador que representa o mecanismo de registro e autenticação do usuário. O nome do arquivo especificado para esta propriedade deve estar em um formato de URL, como "file:///tmp/og/objectGridSecurity.xml".
- `-Dobjectgrid.server.props`: Especifica o arquivo de propriedades do servidor que contém as propriedades de segurança específicas do servidor. O nome do

arquivo especificado para esta propriedade está apenas no formato simples de caminho de arquivo, como ″c:/tmp/og/catalogserver.props″. Observe que o uso de -Dobjectgrid.security.server.props está reprovado, mas ele pode continuar sendo usado para compatibilidade com versões anteriores.

Para iniciar um serviço de catálogo seguro em WebSphere Application Server, siga o ″Incorporado no WebSphere Application Server″ no “Segurança de Grade” na página 286.

Em seguida, a propriedade de segurança no argumento da JVM genérico do processo.

```
-Dobjectgrid.cluster.security.xml.url=file:///tmp/og/
objectGridSecurity.xml-Dobjectgrid.server.props=/tmp/og/
catalog.server.props
```

As etapas para incluir os argumentos da JVM genéricos são as seguintes:

- Expanda ″Administração do sistema″ na visualização de tarefa do lado esquerdo.
- Clique no processo WebSphere Application Server no qual o serviço de catálogo está implementado como, por exemplo, ″Gerenciador de Implementação″.
- Na página à direita, expanda ″Java e Gerenciamento de Processo″ sob ″Infraestrutura do Servidor″.
- Clique em ″Definição de Processo″.
- Clique em ″Java Virtual Machine″ sob ″Propriedades Adicionais″.
- Digite as propriedades na caixa de texto Argumentos JVM Genéricos.

Iniciando um servidor de contêineres seguro no WebSphere Application Server

Um servidor de contêineres, ao se conectar ao servidor de catálogos, obterá todas as configurações de segurança definidas no objectGridSecurity.xml, como a configuração do autenticador ou do tempo limite da sessão de login. Além disso, um servidor de contêineres precisa configurar suas próprias propriedades de segurança específicas do servidor na propriedade -Dobjectgrid.server.props.

É necessário usar a propriedade -Dobjectgrid.server.props em vez da propriedade -Dobjectgrid.security.server.props porque também colocamos outras propriedades não relacionadas à segurança neste arquivo de propriedades. O nome do arquivo especificado para esta propriedade está apenas no formato de caminho de arquivo simples, como `c:/tmp/og/server.props`.

Siga as mesmas etapas acima para incluir a propriedade de segurança para os argumentos genéricos da JVM.

Capítulo 10. Resolução de Problemas

Além dos logs e rastreio, mensagens e notas sobre o release, é possível usar as ferramentas de monitoramento para solucionar problemas em sua configuração.

Usando as Ferramentas de Monitoramento para Resolução de Problemas

Além dos logs e rastreio, mensagens e notas sobre o release, é possível usar as ferramentas de monitoramento para resolver problemas referentes ao local dos dados no ambiente, a disponibilidade dos servidores na grade, etc. Se você estiver executando um ambiente do WebSphere Application Server, poderá usar a Performance Monitoring Infrastructure (PMI). Se você estiver executando em um ambiente independente, poderá usar uma ferramenta de monitoramento do fornecedor, como CA Wily Introscope ou Hyperic HQ. Também é possível usar e customizar o utilitário de amostra xsAdmin para exibir informações de texto sobre o ambiente.

Para obter mais informações sobre as ferramentas de monitoramento, consulte Capítulo 8, “Monitorando seu Ambiente de Implementação”, na página 247.

Logs e Rastreio

É possível utilizar logs e rastreio para monitorar e solucionar problemas em seu ambiente. Os logs estão em locais diferentes dependendo da sua configuração. Pode ser necessário fornecer rastreio para um servidor quando você trabalha com o suporte IBM.

Logs com o WebSphere Application Server

Consulte o WebSphere Application ServerCentro de Informações para obter mais informações.

Logs com o WebSphere eXtreme Scale em um Ambiente Independente

Com os servidores de catálogos e contêineres independentes, é possível configurar o local dos logs e qualquer especificação de rastreio. Os logs do servidor de catálogo estão no local onde você executou o comando do servidor de início.

Configurando o local de log para os servidores de contêiner

Por padrão, os logs para um contêiner estão no diretório onde o comando do servidor foi executado. Se você iniciar os servidores no diretório *<eXtremeScale_home>*/bin, os arquivos de log e de rasteio estão nos diretórios logs/*<server_name>* no diretório bin. Para especificar um local alternativo para os logs do servidor de contêineres, como o arquivo server.properties, com os seguintes conteúdos:

```
workingDirectory=<directory>
traceSpec=
systemStreamToFileEnabled=true
```

A propriedade workingDirectory é o diretório-raiz para os logs e o arquivo de rastreio opcional. O WebSphere eXtreme Scale cria um diretório com o nome do servidor de contêineres com um arquivo SystemOut.log, um arquivo SystemErr.log e um arquivo de rastreio se o rastreio foi ativado com a opção traceSpec. Para usar um arquivo de propriedades durante a inicialização de contêiner, use a opção **-serverProps** e forneça o local do arquivo de propriedades do servidor.

Consulte "Iniciando Servidores WebSphere eXtreme Scale Independentes" na página 218 e "Script startOgServer" na página 223 para obter informações adicionais.

As mensagens de informações comuns a serem procuradas no arquivo SystemOut.log são o início das mensagens de confirmação. Para obter mais informações sobre uma mensagem específica, consulte "Mensagens" na página 309.

Rastreio com o WebSphere Application Server

Consulte o WebSphere Application ServerCentro de Informações para obter mais informações.

Rastreio no Serviço de Catálogo

É possível configurar o rastreio em um serviço de catálogo usando os parâmetros **-traceSpec** e **-traceFile** durante a inicialização do serviço de catálogo. Por exemplo:

```
startOgServer.sh catalogServer -traceSpec
ObjectGridPlacement=all=enabled -traceFile
/home/user1/logs/trace.log
```

Se você iniciar o serviço de catálogo no diretório *<eXtremeScale_home>*/bin, os arquivos de log e de rasteio estarão no diretório logs/*<catalog_service_name>* no diretório bin. Consulte o "Iniciando o Serviço de Catálogo em um Ambiente Independente" na página 218 para obter mais detalhes sobre o início de um serviço de catálogo.

Rastreio em um servidor de contêineres Independente

É possível ativar o rastreio em um servidor de contêiner de duas formas. É possível criar um arquivo de propriedades do servidor conforme explicado na seção de logs ou você pode ativar o rastreio utilizando a linha de comandos na inicialização. Para ativar o rastreio de contêiner com um arquivo de propriedades do servidor, atualize a propriedade **traceSpec** com a especificação de rastreio necessária. Para ativar o rastreio de contêiner utilizando parâmetros de início, utilize os parâmetros **-traceSpec** e **-traceFile**. Por exemplo:

```
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
```

Se você iniciar o servidor no diretório *<eXtremeScale_home>*/bin, os arquivos de log e de rasteio estarão nos diretórios logs/*<server_name>* no diretório bin. Consulte o "Iniciando Processos do Contêiner" na página 221 para obter mais detalhes sobre o início de um processo de contêiner.

Rastreio com a Interface ObjectGridManager

Outra opção é configurar o rastreio durante o tempo de execução em uma interface ObjectGridManager. A configuração de um rastreio em uma interface ObjectGridManager pode ser usada para obter rastreio em um cliente eXtreme

Scale enquanto ele se conecta com um eXtreme Scale e confirma as transações. Para configurar o rastreio em uma interface ObjectGridManager, forneça uma especificação de rastreio e um log de rastreio.

```
ObjectGridManager manager= ObjectGridManagerFactory.getObjectGridManager();
...
manager.setTraceEnabled(true);
manager.setTraceFileName("logs/myClient.log");
manager.setTraceSpecification("ObjectGridReplication=all=enabled");
```

Ativando o Rastreio com o Utilitário xsadmin

Para ativar o rastreio com o utilitário xsadmin, use a opção **setTraceSpec**. Use o utilitário xsadmin para ativar o rastreio em um ambiente independente durante o tempo de execução e não durante a inicialização. É possível ativar o rastreio em todos os servidores e serviços de catálogo ou filtrar os servidores com base no nome do ObjectGrid, e assim por diante. Por exemplo, para ativar o rastreio ObjectGridReplication com acesso ao servidor de serviço de catálogo, execute:

```
<eXtremeScale_home>/bin>xsadmin.bat -setTraceSpec "ObjectGridReplication=all=enabled"
```

Também é possível desativar o rastreio ao configurar a especificação de rastreio para `*=all=disabled.`:

Consulte o “Utilizando o Utilitário de Amostra xsAdmin” na página 266 para obter mais informações.

Diretório e Arquivos ffdc

Os arquivos FFDC servem para que o suporte IBM auxilie na depuração. Estes arquivos podem ser solicitados pelo suporte IBM se ocorrer um problema.

Esses arquivos aparecem no diretório `ffdc` e contêm arquivos semelhantes ao seguinte:

```
server2_exception.log
server2_20802080_07.03.05_10.52.18_0.txt
```

Opções de Rastreio

É possível ativar o rastreio para fornecer informações sobre o seu ambiente para o suporte IBM.

Sobre o Rastreio

O rastreio do WebSphere eXtreme Scale é dividido em vários componentes diferentes. Assim como o rastreio do WebSphere Application Server, é possível especificar o nível de rastreio a ser utilizado. Os níveis comuns de rastreio incluem: all, debug, entryExit e event.

Um exemplo de cadeia de rastreio é o seguinte:

```
ObjectGridComponent=level=enabled
```

É possível concatenar as cadeias de rastreio. Use o sinal de asterisco (`*`) para especificar um valor de curinga, como `ObjectGrid*=all=enabled`. Se for necessário fornecer um rastreio para o suporte IBM, uma cadeia de rastreio específica será solicitada. Por exemplo, se ocorrer um problema com a replicação, a cadeia de rastreio `ObjectGridReplication=debug=enabled` pode ser solicitada.

Especificação de Rastreio

ObjectGrid
Mecanismo de cache principal geral.

ObjectGridCatalogServer
Serviço de catálogo geral.

ObjectGridChannel
Comunicações de topologia de implementação estática.

ObjectgridCORBA
Comunicações de topologia de implementação dinâmica.

ObjectGridDataGrid
A API do AgentManager.

ObjectGridDynaCache
O provedor de cache dinâmico do WebSphere eXtreme Scale.

ObjectGridEntityManager
A API do EntityManager. Utilize com a opção Projector.

ObjectGridEvictors
Evictors integrados do ObjectGrid.

ObjectGridJPA
Carregadores do Java Persistence API (JPA).

ObjectGridJPACache
Plug-ins do Cache JPA

ObjectGridLocking
Gerenciador de bloqueios de entrada de cache do ObjectGrid.

ObjectGridMBean
Beans de gerenciamento.

ObjectGridPlacement
Serviço de disposição de shards do servidor de catálogos.

ObjectGridQuery
Consulte ObjectGrid.

ObjectGridReplication
Serviço de replicação.

ObjectGridRouting
Detalhes de roteamento do cliente/servidor.

ObjectGridSecurity
Rastreio de segurança.

ObjectGridStats
Estatísticas do ObjectGrid.

ObjectGridStreamQuery
API de Consulta do Fluxo.

ObjectGridWriteBehind
Atributo write-behind do ObjectGrid.

Projector
O mecanismo com a API do EntityManager.

QueryEngine
O mecanismo de consulta para a API de Consulta do Objeto e a API de Consulta do EntityManager.

QueryEnginePlan
Diagnósticos do plano de consulta.

Mensagens

Quando encontrar uma mensagem em um log ou em outras partes de uma interface de produto, será possível procurar pela mensagem pelo prefixo do componente para obter mais informações.

Localizando Mensagens

Ao encontrar uma mensagem em um log, copie o número da mensagem com seu prefixo de letra e número e procure no centro de informações (por exemplo, `CWOBJ1526I`). Ao procurar pela mensagem, será possível encontrar uma explicação adicional da mensagem e possíveis ações a serem tomadas para resolver o problema.

Consulte o centro de informações para obter um índice de mensagens do produto.

Notas sobre o Release

São fornecidos links para o Web site de suporte do produto, para documentação do produto e para atualizações de última hora, limitações e problemas conhecidos para o produto.

- "Acessando Últimas Atualizações, Limitações e Problemas Conhecidos"
- "Acessando Requisitos de Software e de Sistema"
- "Acessando a Documentação do Produto"
- "Acessando o Web Site de Suporte do Produto" na página 310
- "Entrando em Contato com o Suporte ao Software IBM" na página 310

Acessando Últimas Atualizações, Limitações e Problemas Conhecidos

As notas sobre o release estão disponíveis como notas técnicas no site de suporte do produto. Para visualizar uma lista de todas as notas técnicas para o WebSphere eXtreme Scale, acesse a página da Web de Suporte.

- Para visualizar uma lista de notas sobre o release para a Versão 7.0, vá para a Página da Web de Suporte.
- Para visualizar uma lista das notas sobre o release para a Versão 6.1, acesse a página wiki das Notas sobre o Release.

Acessando Requisitos de Software e de Sistema

Os requisitos de hardware e software são documentados nas páginas a seguir:

- Requisitos do Sistema Detalhados

Acessando a Documentação do Produto

Para obter o conjunto de informações inteiro, acesse apágina da Biblioteca.

Acessando o Web Site de Suporte do Produto

Para procurar as notas técnicas, downloads, correções e outras informações relacionadas a suporte mais recentes, acesse apágina de Suporte.

Entrando em Contato com o Suporte ao Software IBM

Se você encontrar um problema com o produto, primeiro, tente as seguintes ações:

- Siga as etapas descritas na documentação do produto
- Procure a documentação relacionada na ajuda on-line
- Consulte as mensagens de erro na referência de mensagens

Se você não conseguir resolver o problema com nenhum dos métodos anteriores, entre em contato com o Suporte Técnico IBM.

Desempenho e Boas Práticas

É possível melhorar o desempenho do WebSphere eXtreme Scale.

O WebSphere eXtreme Scale é projetado para melhorar o desempenho do aplicativo. Há muitos fatores que afetam o desempenho do seu aplicativo quando o eXtreme Scale é utilizado. Alguns deles são o tamanho da transação, o modo de cópia, a técnica de serialização e a arquitetura da partição. Para obter o melhor desempenho, você precisa definir a arquitetura e implementar o aplicativo cuidadosamente.

Antes de tentar melhorar o desempenho, consulte as informações sobre transações no *Visão Geral do Produto* para obter mais informações sobre a escolha de uma estratégia de bloqueio para seu ambiente.

Desempenho dos Artefatos do eXtreme Scale

É possível trabalhar com diversos aspectos diferentes do eXtreme Scale para desempenho.

- **Desempenho da serialização:** Consulte as informações sobre o desempenho da serialização no *Visão Geral do Produto* para obter informações adicionais.
- **Tamanho da transação:** Consulte as informações sobre os tamanhos de transação no *Visão Geral do Produto* para obter informações adicionais.
- **Desempenho de replicação:** Consulte as informações sobre o desempenho de replicação no *Visão Geral do Produto* para obter informações adicionais.
- **Impacto do desempenho do gerenciador de entidades:** Consulte as informações sobre o desempenho do gerenciador de entidades no *Guia de Programação* para obter informações adicionais.
- **Ajuste da consulta:** Consulte as informações sobre o ajuste de consultas para desempenho no *Guia de Programação* para obter informações adicionais.

Práticas Recomendáveis

É possível melhorar o desempenho dos mapas com algumas boas práticas. Cada aplicativo e ambiente utiliza uma solução diferente para o desempenho e o eXtreme Scale fornece customizações integradas para melhorar o desempenho. Também é possível melhorar o desempenho na arquitetura do aplicativo. As melhorias são implementadas apenas no contexto do aplicativo e de sua arquitetura.

- **Boas práticas de desempenho de bloqueio**: Escolha entre as estratégias de bloqueio diferentes que afetam o desempenho de seus aplicativos. Consulte as informações sobre as boas práticas de desempenho de bloqueio no *Guia de Programação* para obter mais informações.
- **Boas práticas do método CopyMode**: Escolha entre os diferentes modos de cópia que são usados para alterar como o eXtreme Scale mantém e copia as entradas.Consulte as informações sobre as boas práticas do método CopyMode no *Guia de Programação* para obter mais informações.
- **Boas práticas da interface ObjectTransformer**: Use a interface ObjectTransformer para permitir retornos de chamada para o aplicativo que oferece implementações customizadas de operações comuns e caras, como a serialização de objeto e uma cópia detalhada de um objeto. Consulte as informações sobre as boas práticas da interface ObjectTransformer no *Guia de Programação* para obter mais informações.
- **Boas práticas de desempenho do Evictor do plug-in**: Escolha entre as estratégias de despejo Least Frequently Used (LFU) e Least Recently Used (LRU).Consulte as informações sobre as boas práticas de desempenho do evictor de plug-in no *Guia de Programação* para obter mais informações.
- **Boas práticas do Evictor padrão**: As propriedades para o Evictor time to live (TTL) padrão, que é o evictor padrão criado com cada backingMap.Consulte as informações sobre as boas práticas do evictor padrão no *Guia de Programação* para obter mais informações.
- Ajuste da JVM
- "Usando o WebSphere Real Time" na página 212: Este recurso permite coletas de lixo mais previsíveis junto com um tempo de resposta consistente e estável e rendimentos de transações com o eXtreme Scale.

Monitoramento de Desempenho

O eXtreme Scale também fornece um mecanismo de monitoramento de desempenho. Consulte as seções a seguir para obter estatísticas e ferramentas de monitoramento de desempenho.

- "Módulos PMI" na página 257
- "Monitorando o Desempenho com o PMI do WebSphere Application Server" na página 253

Capítulo 11. Glossário

Este glossário inclui termos e definições para o WebSphere eXtreme Scale.

As referências cruzadas a seguir são utilizadas nesse glossário:

1. Consulte encaminha o leitor para um termo sinônimo preferencial, ou um acrônimo ou uma abreviação para a forma completa definida.
2. Consulte também encaminha o leitor a um termo relacionado ou constrastante.

Para visualizar glossários de outros produtos IBM, vá para www.ibm.com/software/globalization/terminology.

acordo de nível de serviço (SLA). Um contrato entre um cliente e um provedor de serviços que especifica as expectativas para o nível de serviço em relação à disponibilidade, ao desempenho e a outros objetivos mensuráveis.

administrador. Uma pessoa responsável pelas tarefas administrativas como autorização de acesso e gerenciamento de conteúdo. Os administradores também podem alterar os níveis de concessão de autoridade aos usuários.

administrador de segurança. A pessoa que controla o acesso a dados comerciais e funções de programas.

Afinidade de sessão. Um método para configurar aplicativos em que um cliente sempre está conectado ao mesmo servidor. Essas configurações desativam o gerenciamento da carga de trabalho após uma conexão inicial, forçando um pedido do cliente a sempre ir para o mesmo servidor.

Agente. Um programa que executa uma ação em nome de um usuário ou outro programa sem intervenção do usuário ou em um planejamento regular e relata os resultados para o usuário ou programa.

Agente do nó. Um agente administrativo que gerencia todos os servidores de aplicativos em um nó e representa o nó na célula de gerenciamento.

agente iterativo. Uma classe ou construção que é usada para navegar através de uma coleção de objetos, um por vez.

alias de autenticação. Um alias que autoriza acesso aos adaptadores de recursos e origens de dados. Um alias de autenticação contém dados de autenticação, incluindo um ID do usuário e senha.

alta disponibilidade (HA). Pertencente a um sistema em cluster que é reconfigurado quando ocorrem falhas de nó ou de daemon, para que as cargas de trabalho possam ser redistribuídas aos nós restantes no cluster.

ambiente. Uma coleta de recursos lógicos e físicos nomeada usada para suportar o desempenho de uma função.

ambiente de implementação. Um conjunto de clusters, servidores e middleware configurados que colaboram com o fornecimento de um ambiente para hospedar os módulos de software. Por exemplo, um ambiente de implementação pode incluir um host para destinos de mensagem, um processador ou separador de eventos de negócio e programas administrativos.

analista de sistemas. Um especialista responsável pela conversão de requisitos de negócios em definições e soluções do sistema.

APAR. Consulte relatório de análise de programa autorizado.

API. Consulte interface de programação de aplicativos.

API (Application Programming Interface). Uma interface que permite que um programa de aplicativo gravado em uma linguagem de alto nível utilize dados ou funções específicos do sistema operacional ou de um outro programa.

API JavaMail. Uma plataforma e quadro independente do protocolo para construir aplicativos de clientes de correio com base em Java.

aplicativo. Um ou mais programas de computador ou componentes de software que fornecem uma função em suporte direto de um processo de negócios específico ou processos.

aplicativo cliente. Um aplicativo, em execução em uma estação de trabalho e vinculado a um cliente, que fornece ao aplicativo acesso aos serviços de fila em um servidor.

Aplicativo cliente thin. Um tempo de execução de aplicativo Java leve e que pode ser obtido por download capaz de interagir com beans corporativos.

Aplicativo Java EE. Qualquer unidade implementável de funcionalidade Java EE. Esta unidade pode ser um único módulo ou um grupo de módulos empacotados em um arquivo EAR (Enterprise Archive) com um descritor de implementação de aplicativos Java EE. (Sun)

área do editor. Em produtos Eclipse e baseados no Eclipse, a área na janela do ambiente de trabalho em que os arquivos são abertos para edição.

Armazenamento de certificados de coleta. Uma coleta de certificados intermediários ou CRLs (Certificate Revocation Lists) que são utilizadas por um caminho de certificado para construir uma cadeia de certificados para validação.

armazenamento de dados persistente. Um armazenamento não-volátil para dados do evento, como um sistema de banco de dados, que é mantido entre limites de sessão e que continua a existir após a execução do programa ou processo que o criou.

arquivo de armazenamento de dados. Um arquivo de banco de dados de chave que contém as chaves públicas para uma entidade confiável.

arquivo de classe. Um arquivo de origem Java compilado.

arquivo de definição de construção. Um arquivo XML que identifica componentes e características de um pacote de instalação customizada (CIP).

arquivo de exportação.

1. Um arquivo criado durante o processo de desenvolvimento para operações de entrada que contém as definições de configuração para o processamento de entrada.

2. Os dados contendo arquivos que foram exportados.

arquivo de Java. Um formato de arquivo compactado para armazenar todos os recursos necessários para instalar e executar um programa Java em um único arquivo. Consulte também arquivamento Web, archive corporativo.

arquivo delimitado por vírgula. Um arquivo cujos registros contêm campos que são separados por uma vírgula.

Arquivo JAR. Um arquivo archive Java. Consulte também arquivamento Web, archive corporativo.

Arquivo JAR EJB. Um Java Archive que contém um módulo EJB. (Sun)

Arquivo Java. Um arquivo fonte editável (com extensão .java) que pode ser compilado em bytecode (um arquivo .class).

Arquivo JSP. Um arquivo HTML de script que possui uma extensão .jsp e que permite a inclusão de conteúdo dinâmico em páginas da Web. Um arquivo JSP pode ser diretamente solicitado como um URL, chamado por um servlet, ou chamado a partir de uma página HTML.

assíncrono. Relativo a eventos que não estão sincronizados no tempo ou não ocorrem em intervalos de tempo regulares ou previsíveis.

atributo global. No XML, um atributo declarado como um filho do elemento esquema, em vez de parte de uma definição de tipo complexo. Atributos globais podem ser referidos em um ou mais modelos de conteúdo utilizando o atributo ref.

Autenticação. Um serviço de segurança que prova que o usuário de um sistema de computador é de fato quem ele afirma ser. Os mecanismos comuns para implementação deste serviço são as senhas e as assinaturas digitais. A autenticação é diferente da autorização; a autenticação não está relacionada a conceder ou negar acesso aos recursos do sistema.

authorized program analysis report (APAR). Um pedido de correção de um defeito em um release suportado de um programa fornecido pela IBM.

auto-inicialização. Um pequeno programa que carrega programas maiores durante a inicialização do sistema.

autônomos. Independente de qualquer outro dispositivo, programa ou sistema. Em uma rede ou ambiente, uma máquina independente acessa todos os recursos necessários localmente.

Autorização. O processo de concessão a um usuário, sistema ou processo de acesso completo ou restrito a um objeto, recurso ou função.

Balanceamento de carga. A monitoração de servidores de aplicativos e o gerenciamento da carga de trabalho nos servidores. Se um servidor exceder sua carga de trabalho, os pedidos serão encaminhados a outro servidor com mais capacidade.

banco de dados local. Um banco de dados localizado na estação de trabalho em uso.

Bandeja. Um encadeamento que aguarda conexão.

Bean. Uma definição ou instância de um componente JavaBeans. Consulte também JavaBeans, enterprise bean.

Bean corporativo. Um componente que implementa uma tarefa de negócios ou uma entidade de negócios e reside em um contêiner EJB. Beans de entidade, beans de sessão e beans orientados a mensagens são todos beans corporativos. (Sun) Consulte também bean.

bean de comando. Um proxy que pode chamar uma operação única utilizando um método execute().

Bean de entidade. Na programação EJB, um enterprise bean que representa dados persistentes mantidos em um banco de dados. Cada bean de entidade transporta sua própria identidade. (Sun)

Bean Scripting Framework. Uma arquitetura para incorporar funções da linguagem de script a aplicativos Java.

biblioteca.

1. Uma coleta de elementos de modelo, incluindo itens de negócios, processos, tarefas, recursos e organizações.

2. Um projeto que é utilizado para desenvolvimento, gerenciamento de versão, e organização de recursos compartilhados. Apenas um subconjunto de tipos de artefatos pode ser criado e armazenado em uma biblioteca, como objetos de negócios e interfaces.

bifurcação. Um elemento de processo que faz cópias de sua entrada e as redireciona por vários caminhos de processamento paralelamente.

bloqueio. Um meio de evitar que mudanças não confirmadas feitas por um processo aplicativo sejam percebidas por outro processo aplicativo e para evitar que um processo aplicativo atualize dados que estão sendo acessados por outro processo. Um bloqueio garante a integridade dos dados evitando que usuários simultâneos acessem dados inconsistentes.

bloqueio atualizável. Um bloqueio que identifica o intento de atualizar uma entrada armazenada em cache ao usar um bloqueio pessimista.

bloqueio compartilhado. Um bloqueio que limita processos aplicativos que executam simultaneamente às operações somente leitura nos dados do banco de dados.

bloqueio pessimista. Uma estratégia de bloqueio por meio da qual um bloqueio é mantido entre o horário em que uma linha é selecionada e o horário em que uma atualização pesquisada ou operação de exclusão é tentada nesta linha.

bloqueio restrito. Um bloqueio que evita que os processos de aplicativo que executam simultaneamente acessem os dados do banco de dados. Consulte também bloqueio compartilhado.

BMP. Consulte persistência gerenciada por bean.

BMP (Bean-managed Persistence). O mecanismo através do qual a transferência de dados entre as variáveis de um bean de entidade e um gerenciador de recursos é gerenciada pelo bean de entidade. (Sun)

BMT. Consulte transação gerenciada por bean.

BMT (Transação Gerenciada por Bean). O recurso do bean de sessão, servlet ou componente do aplicativo cliente que gerencia suas próprias transações diretamente, em vez de fazê-lo através de um contêiner.

bootstrapping. O processo pelo qual uma referência inicial do serviço de nomenclatura é obtido. A definição de bootstrap e o nome do host formam o contexto inicial das referências de JNDI (Java Naming and Directory Interface).

bytecode. Código independente da máquina gerado pelo compilador Java e executado pelo interpretador Java. (Sun)

cache coerente. O cache que mantém a integridade para que todos os clientes vejam os mesmos dados.

Cache dinâmico. A consolidação de várias atividades de cache, inclusive servlets, serviços da Web e comandos do WebSphere em um serviço no qual essas atividades compartilham parâmetros de configuração e funcionam juntas para melhorar o desempenho.

cache read-through. Um cache escasso que carrega entradas de dados por chave à medida que são solicitadas. Quando os dados não podem ser localizados no cache, os dados ausentes são recuperados com o utilitário de carga, que carrega os dados do repositório de dados de backend e insere os dados no cache.

cache write-behind. Um cache que grava assincronicamente cada operação de gravação para o banco de dados usando um utilitário de carga.

cache write-through. Um cache que grava sincronicamente cada operação de gravação para o banco de dados usando um utilitário de carga.

Caminho de classe. Uma lista de diretórios e arquivos JAR que contém os arquivos de recursos ou classes Java que um programa pode carregar dinamicamente no tempo de execução.

caminho de construção. O caminho utilizado durante a compilação do código fonte Java para localizar classes referenciadas que residem em outros projetos.

Canal de contêiner da Web. Um tipo de canal dentro de uma cadeia de transporte que cria uma ponte na cadeia de transporte entre um canal de entrada HTTP e um servlet ou mecanismo JSP (JavaServer Pages).

Canal SSL. Um tipo de canal dentro de uma cadeia de transporte que associa um repertório de configuração SSL (Secure Sockets Layer) com a cadeia de transporte.

Canal TCP. Um tipo de canal dentro de uma cadeia de transporte que fornece aplicativos clientes com conexões persistentes em uma LAN (Rede Local).

carregador de classes. Parte da JVM (Java Virtual Machine) que é responsável por localizar e carregar arquivos de classe. Um carregador de classes afeta o pacote dos aplicativos e o comportamento do tempo de execução dos aplicativos empacotados implementados em servidores de aplicativos.

catálogo. Um contêiner que, dependendo do tipo de contêiner, suspende processos, dados, recursos, organizações ou relatórios na árvore de projeto.

categoria. Um contêiner usado em um diagrama de estrutura para agrupar elementos do grupo com base em um atributo compartilhado ou qualidade.

cell.

1. Um grupo de processos gerenciados que são associados ao mesmo gerenciador de implementação e podem incluir grupos principais de alta disponibilidade.

2. Um ou mais processos, cada com componentes de tempo de execução de host. Cada uma tem um ou mais grupos principais denominados.

centro de informações. Uma coleta de informações que fornece suporte para os usuários de um ou mais produtos pode ser ativada separadamente do produto e inclui uma lista de tópicos para navegação e um mecanismo de procura.

Certificado de assinante. A entrada do certificado confiável que normalmente está em um arquivo de armazenamento confiável.

Certificado digital. Um documento eletrônico usado para identificar um indivíduo, um sistema, um servidor, uma empresa ou alguma outra entidade e para associar uma chave pública à entidade. Um certificado digital é emitido por uma autoridade de certificação e é assinado digitalmente por essa autoridade.

chamada. A ativação de um programa ou procedimento.

chassi. A estrutura de metal na qual vários componentes eletrônicos são montados.

chave.

1. Um valor matemático criptografado que é utilizado para atribuir digitalmente, verificar, criptografar ou decriptografar uma mensagem.

2. Informações que caracterizam e identificam de forma exclusiva a entidade de mundo real que está sendo rastreada por um contexto de monitoramento.

Chave primária.

1. Um objeto que identifique unicamente um bean de entidade de um tipo específico.

2. Em um banco de dados relacional, uma chave que identifica exclusivamente uma linha de uma tabela de banco de dados.

ciclo de vida. Uma passagem completa pelas quatro fases de desenvolvimento de software: encetamento, elaboração, construção e transição.

CIP. Consulte o pacote de instalação customizada.

classe. Na programação ou no design orientado pelo objeto, um modelo ou modelo que pode ser utilizado para criar objetos com uma definição, propriedades, operações e comportamento comuns. Um objeto é uma instância de uma classe.

classe de bean. Na programação EJB (Enterprise JavaBeans), uma classe Java que implementa uma classe javax.ejb.EntityBean ou classe javax.ejb.SessionBean.

Classe Java. Uma classe que é gravada na linguagem Java.

classificador. Um atributo especializado usado para agrupamento e processo de codificação por cor dos elementos.

cliente. Um programa de software ou computador que solicita serviços de um servidor. Consulte também host.

Cliente fino. Um cliente com pouco ou nenhum software instalado, mas que tem acesso a software gerenciado e fornecido por servidores de rede conectados a ele. Um cliente thin é uma alternativa a um cliente com plena funcionalidade, como um estação de trabalho.

cliente/servidor. Pertencente ao modelo de interação em processamento de dados distribuídos, no qual um programa em um computador envia um pedido para um programa em outro computador e aguarda uma resposta. O programa solicitante é chamado de cliente; o programa de resposta é chamado de servidor.

Cloudscape. Um ORDBMS (Object-Relational Database Management System) incorporável totalmente Java.

cluster. Um grupo de servidores de aplicativos que colabora com o equilíbrio de carga de trabalho e failover.

Cluster de proxy. Um grupo de servidores de proxy que distribui pedidos HTTP através do cluster.

Cluster de servidores. Um grupo de servidores que geralmente estão em máquinas físicas diferentes e que possuem os mesmos aplicativos neles configurados, mas que operam como um único servidor lógico.

cluster dinâmico. Um cluster de servidor que utiliza pesos para equilibrar as cargas de trabalho de seus membros de cluster dinamicamente, com base nas informações de desempenho coletadas dos membros do cluster.

código de implementação. Código adicional que permite que o código de implementação do bean gravado por um desenvolvedor de aplicativos funcione em um determinado ambiente de tempo de execução EJB. O código de implementação pode ser gerado por ferramentas fornecidas pelo fornecedor de servidor de aplicativos.

coleta de lixo. Uma rotina que pesquisa a memória para reclamar espaço de segmentos do programa ou dados inativos.

componente.

1. Um objeto reutilizável ou programa que executa uma função específica e trabalha com outros componentes e aplicativos.

2. No Eclipse, um ou mais plug-ins que funcionam em conjunto para fornecer um conjunto distinto de funções.

Componente da Web. Um servlet, arquivo JSP (JavaServer Pages), ou arquivo HTML (HyperText Markup Language). Um ou mais componentes da Web formam um módulo da Web.

conflito. Uma condição na qual dois encadeamentos de controle independentes estão bloqueados, cada um aguardando pelo outro tomar uma ação. O conflito geralmente surge da inclusão de mecanismos de sincronização para evitar condições de disputa.

Consulta EJB. Na linguagem de consulta EJB, uma cadeia que contém uma cláusula SELECT opcional, que especifica os objetos EJB a serem retornados, uma cláusula FROM, que nomeia as coletas de beans, uma cláusula WHERE opcional, que contém predicados de procura nas coletas e uma cláusula ORDER BY opcional, que especifica a ordem da coleta de resultados e dos parâmetros de entrada que correspondem aos argumentos do método localizador.

consultar.

1. Uma solicitação de informações de um banco de dados baseada em condições específicas: por exemplo, uma solicitação para uma lista de todos os clientes em uma tabela de clientes cujos saldos sejam maiores que USD1000.

2. Um pedido reutilizável de informações sobre um ou mais elementos de modelo

consulta SQL. Um componente de determinadas instruções SQL que especifica uma tabela de resultados.

Contêiner da Web. Um contêiner que implementa o contrato do componente Web da arquitetura Java EE. (Sun)

Contêiner de EJB. Um contêiner que implementa o contrato do componente EJB da arquitetura Java EE. Esse contrato especifica um ambiente de tempo de execução para os beans corporativos que incluem segurança, conformidade, gerenciamento do ciclo de vida, transação, implementação e outros serviços. (Sun)

contenção de encadeamento. Uma condição na qual um encadeamento está aguardando uma trava ou um objeto que um outro encadeamento possui.

Contexto EJB. Nos beans corporativos, um objeto que permite a um enterprise bean chamar serviços fornecidos pelo contêiner e para obter informações sobre o responsável pela chamada de um método chamado pelo cliente. (Sun)

conversor. Na programação EJB (Enterprise JavaBeans), uma classe que converte uma representação do banco de dados em um tipo de objeto e vice-versa.

correção temporária. Uma correção certificada que geralmente está disponível para todos os clientes entre fix packs, pacotes de atualização ou releases regularmente planejados. Consulte também fix pack.

crawler da Web. Um tipo de crawler que explora a Web recuperando um documento da Web e seguindo os links nesse documento.

Credencial. No quadro JAAS (Java Authentication and Authorization Service), uma classe de assunto que possui atributos relacionados à segurança. Esses atributos podem conter informações utilizadas para autenticar o assunto para novos serviços.

cronômetro. Uma tarefa que produz saída em determinados pontos no tempo.

Customized Installation Package (CIP). Uma imagem de instalação customizada que pode incluir um ou mais pacotes de manutenção, um arquivo arquivado de configuração a partir de um perfil de servidor independente, um ou mais arquivos arquivados corporativos, scripts e outros arquivos que ajudam a customizar a instalação resultante.

dados da hora de construção. Objetos que não são utilizados pelo conversor, como padrões EDI, tipos de documentos de dados orientados por registros e mapas.

daemon. Um programa que é executado de modo não assistido para executar funções contínuas ou periódicas, como controle de rede.

DB2. Uma família de programas licenciados da IBM para gerenciamento de banco de dados relacional.

definição de documento DTD. Uma descrição ou um layout de um documento XML baseado em um DTD XML.

demilitarized zone (DMZ). Uma configuração que inclui múltiplos firewalls para incluir camadas de proteção entre uma intranet corporativa e uma rede pública, como a Internet.

Depósito de informações do provedor. Na programação orientada a objetos, uma classe que é utilizada para criar instâncias de uma outra classe. Uma fábrica é utilizada para isolar a criação de objetos de uma determinada classe em um local para que novas funções possam ser fornecidas sem difundir as alterações do código.

derivação. Na programação orientada a objetos, o refinamento ou a extensão de uma classe em outra.

descoberta automática. A descoberta de artefatos de serviço em um sistema de arquivos, registro externo ou outra origem.

Descrição, Descoberta e Integração Universal (UDDI). Um conjunto de especificações baseadas em padrões que permite às companhias e aos aplicativos localizarem e utilizarem os serviços da Web na Internet, de maneira rápida e fácil.

Descritor de implementação. Um arquivo XML que descreve como implementar um módulo ou aplicativo, especificando opções de configuração e de contêiner. Por exemplo, um descritor de implementação EJB transmite informações para um contêiner de EJB sobre como gerenciar e controlar um enterprise bean.

desenvolvimento de baixo para cima. Nos serviços da Web, o processo de desenvolvimento de um serviço com base em um artefato existente, como um Java bean ou enterprise bean, em vez de um arquivo WSDL (Web Services Description Language).

desserialização. Um método para converter uma variável serializada em dados de objetos.

destino. Um ponto de saída usado para fornecer entregar documentos a um sistema backend ou a um parceiro comercial.

destino de instalação. O sistema no qual os pacotes de instalação selecionados são instalados.

diretório de implementação. O diretório em que a configuração de servidor publicada e o aplicativo da Web estão localizados na máquina em que o servidor de aplicativos está instalado.

diretório LDAP. Um tipo de repositório que armazena informações sobre pessoas, organizações e outros recursos e que é acessado utilizando o protocolo LDAP. As entradas do repositório são organizadas em uma estrutura hierárquica e, em alguns casos, a estrutura hierárquica reflete a estrutura ou a geografia de uma organização.

disparar. Na programação orientada por objetos, causar uma transição de estado.

disponibilidade.

1. A condição que permite aos usuários acessar e usar seus aplicativos e dados.

2. Os períodos de tempo durante os quais um recurso está acessível. Por exemplo, um contratado pode ter uma disponibilidade das 9h às 17h, todos os dias da semana e das 9h às 15h aos sábados.

DMZ. Consulte zona desmilitarizada.

DNS. Consulte Domain Name System.

Document Type Definition (DTD). As regras que especificam a estrutura de uma classe específica de documentos SGML ou XML. O DTD define a estrutura com elementos, atributos e notações e estabelece restrições para como cada elemento, atributo e notação pode ser utilizado na classe de documentos específica.

domain. Um objeto, ícone ou contêiner que contém outros objetos que representam os recursos de um domínio. O objeto de domínio pode ser utilizado para gerenciar esses recursos.

Domain Name System (DNS). O sistema de banco de dados distribuído que mapeia nomes de domínio para endereços IP.

Drop-down. Consulte pull-down.

DTD. Consulte definição de tipo de documento.

EAR. Consulte archive corporativo.

EAR (Enterprise Archive). Um tipo especializado de arquivo JAR, definido pelo padrão Java EE, utilizado para implementar aplicativos Java EE em servidores de aplicativos do Java EE. Um arquivo EAR contém componentes do EJB, um descritor de implementação e arquivos WAR (Web Archive) para aplicativos individuais da Web. Consulte também arquivamento Web.

Eclipse. Uma iniciativa de código aberto que fornece a ISVs e a outros desenvolvedores de ferramentas uma plataforma padrão para o desenvolvimento de ferramentas de desenvolvimento de aplicativos compatíveis com conexão.

edição. Uma geração de implementação sucessiva de um conjunto particular de artefatos com versão.

EJB. Consulte JavaBeans Corporativos.

elemento do componente. Uma entidade em um componente, na qual um ponto de interrupção pode ser definido, como uma atividade ou snippet Java em um processo de negócios, ou uma primitiva ou nó de mediação em um fluxo de mediação.

elemento global. No XML, um elemento declarado como um filho do elemento esquema, em vez de parte de uma definição de tipo complexo. Os elementos globais podem ser referidos em um ou mais modelos de conteúdo, utilizando o atributo ref.

encadeamento. Um fluxo de instruções do computador que está no controle de um processo. Em alguns sistemas operacionais, um encadeamento é a menor unidade de operação em um processo. Vários encadeamentos podem ser executados simultaneamente, executando tarefas diferentes.

Enterprise JavaBeans (EJB). Uma arquitetura de componente definida pela Sun Microsystems para o desenvolvimento e a implementação de aplicativos orientados por objetos, distribuídos e de nível corporativo (Java EE).

enterprise service bus (ESB). Uma infra-estrutura de conectividade flexível para integração de aplicativos e serviços; oferece uma abordagem flexível e gerenciável para a implementação de arquitetura orientada a serviços.

Entidade.

1. Uma classe Java simples que representa uma linha em uma tabela de banco de dados ou entrada em um mapa.

2. Em linguagens de marcação como XML, uma coleção de caracteres que podem se referenciados como uma unidade, por exemplo, para incorporar texto frequentemente repetido ou caracteres especiais dentro de um documento.

envio de dados. Relativo à direção do fluxo, que é do início do processo (envio de dados) para o fim do processo (recebimento de dados).

Erro. Uma discrepância entre um valor ou condição calculada, observada ou medida e o valor ou condição verdadeira, especificada ou teoricamente correta.

ESB. Consulte barramento de serviço corporativo.

escalabilidade. A capacidade de um sistema expandir quando recursos, como processadores, memória ou armazenamento, são incluídos.

Escopo.

1. Uma especificação do limite dentro do qual os recursos do sistema podem ser usados.

2. Em serviços da Web, uma propriedade que identifica o tempo de vida do objeto que atende a solicitação de chamada.

espaço de nomes. Um contêiner lógico no qual todos os nomes são exclusivos. O identificador exclusivo para um artefato é composto do espaço de nomes e do nome local do artefato.

espaço de trabalho.

1. Um diretório no disco que contém todos os arquivos de projeto, assim como as informações como preferências.

2. Um repositório temporário de informações de configuração que clientes administrativos utilizam.

. No Eclipse, o conjunto de projetos e de outros recursos que o usuário está desenvolvendo no momento no ambiente de trabalho. Metadados sobre este recursos residem em um diretório no sistema de arquivos; os recursos podem residir no mesmo diretório.

específico. Relativo à visualização de um objeto individual em detalhes.

esqueleto. Estrutura para uma classe de implementação.

Esquema de URL. Um formato que contém outra referência de objeto.

Estático. Uma palavra-chave da linguagem de programação Java utilizada para definir uma variável como variável de classe.

Evento.

1. Uma mudança para um estado, como a conclusão ou falha de uma operação, processo de negócios ou tarefa manual, que pode disparar uma ação subsequente, como persistir dados do evento para um repositório de dados ou chamar outro processo de negócios.

2. Uma mudança nos dados em um enterprise information system (EIS) que é processada pelo adaptador e usada para fornecer objetos de negócios do EIS aos terminais (aplicativos) que precisam ser notificados da mudança.

evictor. Um componente que controla a associação de entradas em cada instância de BackingMap. Caches escassos podem usar evictors para remover dados automaticamente do cache sem afetar o banco de dados.

exceção. Uma condição ou evento não pode ser tratada por um processo normal.

exportação. Uma interface exposta de um módulo SCA (Service Component Architecture) que oferece um serviço de negócios externamente. Uma exportação tem uma ligação que define como o serviço pode ser acessado por solicitantes de serviços, por exemplo, um serviço da Web.

expressão. Um operando SQL ou XQuery ou uma coleta de operadores e operandos SQL ou XQuery que produzem um único valor.

Extensible Markup Language (XML). Uma metalinguagem padrão para definir linguagens de marcações com base em SGML (Standard Generalized Markup Language).

eXtreme Scale distribuído. Um padrão de uso para interagir com o eXtreme Scale quando os servidores e os clientes existem em vários processos.

factory EJB. Um bean de acesso que simplifica a criação ou a descoberta de uma instância de enterprise bean.

failover. Uma operação automática que comuta para um sistema redundante ou de espera no caso de uma interrupção de software, hardware ou rede.

fase de implementação. Consulte também fase de implementação.

fase de implementação. Uma fase que inclui uma combinação de criação de ambiente de hosting para seus aplicativos e a implementação desses aplicativos. Isso inclui resolver as dependências de recurso do aplicativo, condições operacionais, requisitos de capacidade e limitadores de integridade e acesso.

Firewall. Uma configuração de rede, geralmente de hardware e software, que impede que um tráfego não autorizado entre e saia de uma rede segura.

Fixpack. Uma coleta acumulativa de correções que é disponibilizada entre pacotes de atualização, atualizações de fábrica ou releases planejados. Ela é destinada a permitir que os usuários se movam para um nível de manutenção específico. Consulte também correção temporária.

Fluxo do log de erros. Um fluxo contínuo de informações de erro que é transmitido utilizando um formato predefinido.

formato binário. Representação de um valor decimal no qual cada campo deve ter 2 ou 4 bytes de comprimento. O sinal (+ ou -) está no bit mais à esquerda do campo e o valor de número está nos bits restantes do campo. Números positivos têm um 0 no bit de sinal e estão na forma verdadeira. Os números negativos têm um 1 no bit de sinal e estão em duas formas de complemento.

fuga de memória. O efeito de um programa que mantém referências a objetos que não são mais necessários e, portanto, precisam ser recuperados.

funcionamento. A condição ou estado geral do ambiente do banco de dados.

Função.

1. Uma descrição de uma função a ser desempenhada por um indivíduo ou recurso em massa, e as qualificações necessárias para preencher a função. Na simulação e na análise, a função do termo também é usada para se referir aos recursos qualificados.

2. Um cargo que identifica as tarefas que um usuário pode executar e os recursos aos quais um usuário possui acesso. Um usuário pode ter uma ou mais funções atribuídas.

. Um grupo lógico de diretores que fornece um conjunto de permissões. O acesso a operações é controlado pela concessão de acesso a uma função.

4. Em um relacionamento, uma função determina a função e a participação de entidades. As funções capturam os requisitos de estrutura e de restrição nas entidades participantes e sua maneira de participação. Por exemplo, em um relacionamento de emprego, as funções são empregador e funcionário.

gargalo. Um local no sistema no qual a contenção para um recurso está afetando o desempenho.

genérico. Em relação à visualização de um grupo de objetos com base em um resumo ou alto nível.

gerenciador autônomo. Um conjunto de componentes de software ou hardware, configurado por políticas, as quais gerenciam o comportamento de outros componentes de software ou hardware como um humano o faria. Um gerenciador autônomo inclui um loop de controle que consiste em componentes de monitor, análise, plano e execução.

Gerenciador de alta disponibilidade. Um quadro dentro do qual a participação do grupo principal é determinado e o status é comunicado entre os membros do grupo principal.

gerenciador de implementação. Um servidor que gerencia operações para um grupo lógico de células de outros servidores.

gerenciamento de carga de trabalho. A otimização da distribuição de pedidos de trabalho de entrada para os servidores de aplicativos, beans corporativos, servlets e outros objetos que podem efetivamente processar o pedido.

GIOP. Consulte General Inter-ORB Protocol.

GIOP (General Inter-ORB Protocol). Um protocolo que a arquitetura CORBA (Common Object Request Broker Architecture) utiliza para definir o formato de mensagens.

global.

1. Pertinente a um elemento que está disponível a qualquer processo na área de trabalho. Um elemento global aparece na árvore de projeto e pode ser usado em vários processos. Tarefas, processos, repositórios e serviços podem ser globais (com referência de qualquer processo no projeto) ou locais (específicas para um único processo).

2. Pertinente às informações disponíveis a mais de um programa ou subrotina.

grade de dados. Um sistema para acessar terabytes ou petrabytes de dados.

grade de eXtreme Scale. Um padrão que é usado para interagir com o eXtreme Scale quando todos os dados e clientes estão em um processo.

grupo.

1. Uma coleta de usuários que podem compartilhar autoridades de acesso para recursos protegidos.

2. Um conjunto de documentos relacionados dentro de um intercâmbio. Um intercâmbio pode conter de zero a vários grupos.

. Em locais, duas ou mais pessoas que são agrupadas para associação em um local.

grupo HA. Um conjunto de um ou mais membros usados para fornecer alta disponibilidade para um processo.

HA. Consulte alta disponibilidade.

Herança. Uma técnica de programação orientada a objetos na qual as classes existentes são usadas como base para criar outras classes. Por intermédio da herança, os elementos mais específicos incorporam a estrutura e o comportamento de elementos mais gerais.

Herança EJB. Uma forma de herança na qual um enterprise bean herda propriedades, métodos e atributos do descritor de controle do nível do método de outro enterprise bean que reside no mesmo grupo.

Hierarquia de classe. As relações entre as classes que compartilham uma herança única.

Host.

1. Um computador que está conectado a uma rede e que fornece um ponto de acesso para essa rede. O host pode ser um cliente, um servidor ou ambos um cliente e servidor simultaneamente.

2. Na definição do perfil de desempenho, uma máquina que possui processos cujos perfis estão sendo definidos. Consulte também servidor.

Host virtual. Uma configuração que permite que uma única máquina host se pareça com várias máquinas host. Os recursos associados a um host virtual não podem compartilhar dados com os recursos associados a um outro host virtual, mesmo que os hosts virtuais compartilharem da mesma máquina física.

HTTP através de SSL (HTTPS). Um protocolo da Web para transações seguras que criptografa e decriptografa pedidos de páginas do usuário e páginas retornadas pelo servidor da Web.

HTTPS.

1. Consulte HTTP sobre SSL.

2. Consulte Segurança de Protocolo de Transporte de Hipertexto.

Hypertext Transfer Protocol Secure (HTTPS). Um protocolo da Internet que é utilizado por servidores da Web e navegadores da Web para transferir e exibir documentos de hipermídia com segurança através da Internet.

IDE. Consulte ambiente de desenvolvimento integrado.

IDE (Integrated Development Environment). Um conjunto de ferramentas de desenvolvimento de software como editores de origem, compiladores e depuradores, que são acessíveis a partir de uma interface com o usuário única.

identificador de instância global. Um identificador globalmente exclusivo que é gerado pelo aplicativo ou pelo emissor e é utilizado como uma chave primária para identificação de eventos.

IIOP. Consulte Internet Inter-ORB Protocol.

IIOP (Internet Inter-ORB Protocol). Um protocolo utilizado para a comunicação entre intermediários de pedidos de objetos CORBA (Common Object Request Broker Architecture).

implementar. Colocar arquivos ou instalar software em um ambiente operacional. No J2EE (Java 2 Platform, Enterprise Edition), essa ação envolve a criação de um descritor de implementação adequado ao tipo de aplicativo que está sendo implementado.

importação.

1. Um artefato de desenvolvimento que importa um serviço que é externo a um módulo.

2. O ponto no qual um módulo SCA acessa um serviço externo, (um serviço externo ao módulo SCA) como se ele fosse local. Uma importação define as interações entre o módulo SCA e o provedor de serviços. Uma importação possui uma ligação e uma ou mais interfaces.

índice. Um conjunto de ponteiros que são logicamente ordenados pelos valores de uma chave. Os índices fornecem rápido acesso aos dados e podem impingir exclusividade dos valores da chave às linhas na tabela.

instalação silenciosa. Uma instalação que não envia mensagens ao console, mas que armazena mensagens e erros nos arquivos de log. Uma instalação silenciosa pode utilizar arquivos de resposta para a entrada de dados.

instância. Uma ocorrência específica de um objeto que pertence a uma classe.

instância do componente. Um componente em execução, que pode estar em execução paralelamente a outras instâncias do mesmo componente.

instanciar. Representar uma abstração com uma instância concreta.

interface. Um conjunto de operações usadas para especificar um serviço de uma classe ou um comportamento.

IP. Consulte Protocolo da Internet.

IP sprayer. Um dispositivo localizado entre os pedidos de entrada dos usuários e os nós do servidor de aplicativos que roteia novamente os pedidos através dos nós.

iteração. Consulte loop.

JAAS. Consulte Java Authentication and Authorization Service.

JAAS (Java Authentication and Authorization Service). Na tecnologia Java EE, uma API padrão para executar operações baseadas em segurança. Através de JAAS, os serviços podem autenticar e autorizar usuários enquanto ativam os aplicativos para permanecer independentes das tecnologias subjacentes.

JAF. Consulte JavaBeans Activation Framework.

JAF (JavaBeans Activation Framework). Uma extensão padrão para a plataforma Java que determina os tipos de dados arbitrários e operações disponíveis e que pode instanciar um bean para executar serviços pertinentes.

(Java. Uma linguagem de programação orientada a objetos para código interpretativo portátil que suporta a interação entre objetos remotos. Java foi desenvolvido e especificado por Sun Microsystems, Incorporated.

JavaBeans. Conforme definido para Java pela Sun Microsystems, um modelo de componente portátil, reutilizável, independente da plataforma. Consulte também bean.

Java Command Language. Uma linguagem de script para o ambiente Java utilizada para criar conteúdo da Web e controlar aplicativos Java.

Javadoc.

1. Uma ferramenta que analisa as declarações e os comentários da documentação em um conjunto de arquivos de origem e produz um conjunto de páginas HTML que descreve as classes, classes internas, interfaces, construções, os métodos e campos. (Sun)

2. Pertinente à ferramenta que analisa as declarações e os comentários da documentação em um conjunto de arquivos de origem e produz um conjunto de páginas HTML que descreve as classes, classes internas, interfaces, construtores, os métodos e campos.

Java EE. Consulte Plataforma Java, Enterprise Edition.

Java EE Connector Architecture (JCA). Uma arquitetura padrão para conectar a plataforma Java EE ao enterprise information systems (EIS) heterogêneo.

Java Message Service (JMS). Uma interface de programação de aplicativo que fornece funções de linguagem Java para manipulação de mensagens.

Java Platform, Enterprise Edition (Java EE). Um ambiente para desenvolvimento e implementação de aplicativos corporativos, definido pela Sun Microsystems Inc. A plataforma Java EE consiste em um conjunto de serviços, APIs (interface de programação de aplicativos) e protocolos que fornecem a funcionalidade para desenvolver aplicativos com multicamadas, baseados na Web. (Sun)

Java runtime environment. Um subconjunto do Java developer kit que contém os programas principais executáveis e os arquivos que constituem a plataforma Java padrão. O JRE inclui a JVM (Java Virtual Machine), classes de núcleo e arquivos de suporte.

JavaScript. Uma linguagem de script da Web utilizada em navegadores e servidores da Web. (Sun)

JavaScript Object Notation. Formato de troca de dados simples baseado na notação de objeto literal do JavaScript. JSON é uma linguagem de programação neutra, porém utilizando convenções de linguagens que incluem C, C++, C#, Java, JavaScript, Perl, Python.

Java SE. Consulte Java Platform, Standard Edition.

Java SE Development Kit (JDK). O nome do kit de desenvolvimento de software fornecido pela Sun Microsystems para a plataforma Java.

Java Specification Request (JSR). Uma especificação proposta formalmente para a plataforma Java.

Java virtual machine Profiler Interface (JVMPI). Uma ferramenta de perfil com suporte à coleta de informações, como dados sobre coleta de lixo e a API da Java virtual machine (JVM) que executa o servidor de aplicativos.

JAX. Consulte Java API for XML.

JAX (Java API para XML). Um conjunto de APIs baseadas em Java para manipular várias operações envolvendo dados definidos através de XML (Extensible Markup Language).

JCA. Consulte Java EE Connector Architecture.

JDBC. Consulte Java Database Connectivity.

JDBC (Java Database Connectivity). Um padrão de mercado para conectividade independente de banco de dados entre a plataforma e um amplo intervalo de bancos de dados. A interface JDBC fornece uma interface de nível de chamada para acesso ao banco de dados baseado em SQL e baseado em XQuery.

JDK. Consulte Java SE Development Kit.

JMS. Consulte Java Message Service.

JMX. Consulte Java Management Extensions.

JMX (Java Management Extensions). Um meio de executar gerenciamento e através de tecnologia Java. JMX é uma extensão universal aberta da linguagem de programação Java para gerenciamento que pode ser implementada entre todas as indústrias, onde o gerenciamento for necessário.

JNDI. Consulte Java Naming and Directory Interface.

JNDI (Java Naming and Directory Interface). Uma extensão para a plataforma Java que fornece uma interface padrão para serviços de nomenclatura heterogêneas e de diretório.

JSP. Consulte JavaServer Pages.

JSP (JavaServer Pages). Uma tecnologia de script no lado do servidor que permite que código Java seja incorporado dinamicamente dentro de páginas da Web (arquivos HTML) e executado quando a página for apresentada, para retornar conteúdo dinâmico ao cliente.

JSR. Consulte Java Specification Request.

JSSE. Consulte Java Secure Socket Extension.

JSSE (Java Secure Socket Extension). Um pacote Java que permite comunicações seguras na Internet. Ele implementa uma versão Java dos protocolos SSL (Secure Sockets Layer) e TLS (Transport Layer Security) e suporta criptografia de dados, autenticação de servidor, integridade de mensagens e autenticação de cliente opcional.

junção.

1. Um elemento de processo que recombina e sincroniza os caminhos de processamento paralelo depois de uma decisão ou bifurcação. Uma junção aguarda a chegada de uma entrada em cada uma de suas ramificações antes de permitir que o processo continue.

2. Uma operação relacional SQL na qual os dados podem ser recuperados de duas tabelas, tipicamente com base em uma condição de junção especificando as colunas de junção.

3. A configuração em um link de recebimento que determina o comportamento do link.

JVM. Consulte Java virtual machine.

JVM (Java Virtual Machine). Uma implementação de software de um processador que executa código Java compilado (applets e aplicativos).

JVMPI. Consulte Java virtual machine Profiler Interface.

Jython. Uma implementação da linguagem de programação Python integrada com a plataforma Java.

kit de desenvolvimento de software (SDK). Um conjunto de ferramentas, APIs e documentação para assistir no desenvolvimento do software em uma linguagem de computador específica ou para um ambiente operacional particular.

LDAP. Consulte Lightweight Directory Access Protocol.

LDAP (Lightweight Directory Access Protocol). Um protocolo baseado em padrão aberto que utiliza o TCP/IP para fornecer acesso a diretórios que suportam um modelo X.500 e que não está sujeito aos requisitos de recursos do DAP (Directory Access Protocol) X.500 mais complexo. Por exemplo, o LDAP pode ser utilizado para localizar pessoas, organizações e outros recursos em um diretório da Internet ou da intranet.

LED com login. Conforme definido em um documento WSDL (Web Services Description Language), um único nó de extremidade que é definido como uma combinação de uma ligação e um endereço de rede.

leitura suja. Um pedido de leitura que não envolve nenhum mecanismo de bloqueio. Isso significa que os dados que podem ser lidos podem ser recuperados posteriormente, resultando em uma inconsistência entre o que foi lido e o que está no banco de dados.

Ligação com escopo em célula. Um escopo de ligação no qual a ligação não é específica e não está associada a nenhum nó ou servidor. Esse tipo de ligação de nomes é criado no contexto de raiz persistente de uma célula.

ligação de dados JMS. Uma ligação de dados que fornece um mapeamento entre o formato utilizado por uma mensagem JMS externa e a representação SDO (Service Data Object) utilizada por um módulo SCA (Service Component Architecture).

ligação de protocolo. Uma ligação que permite que o barramento de serviço corporativo processe mensagens, independentemente do protocolo de comunicação.

limite. Uma configuração que se aplica a uma interrupção em uma simulação que define quando uma simulação de processo deve ser parada com base em uma condição existente para uma proporção especificada de ocorrências de algum evento.

linha de comandos. A linha em branco em uma exibição em que os comandos, os números de opção ou as seleções podem ser digitadas.

Listener. Um programa que detecta pedidos de entrada e inicia o canal associado.

listener do terminal. O ponto ou endereço no qual as mensagens que chegam para um serviço da Web são recebidas por um barramento de integração de serviço.

Local.

1. Relativo a um dispositivo, arquivo ou sistema que é acessado diretamente de um sistema do usuário, sem o uso de uma linha de comunicação.

2. Relativo a um elemento que está disponível somente em seu próprio processo.

log. O registro de dados sobre eventos específicos no sistema, como erros.

loop. Uma seqüência de instruções desempenhadas repetidamente.

loop do-while. Um loop que repete a mesma seqüência de atividades enquanto uma condição é atendida. Diferentemente de um loop while, um loop do-while testa sua condição no final do loop. Isso significa que sua seqüência de atividades sempre é executada pelo menos uma vez.

loop for. Um loop que repete a mesma seqüência de atividades por um determinado número de vezes.

loop while. Um loop que repete a mesma seqüência de atividades enquanto uma condição é atendida. O loop while testa sua condição no início de cada loop. Se a condição for false desde o início, a seqüência de atividades contida no loop nunca será executada.

LTPA. Consulte Lightweight Third Party Authentication.

LTPA (Lightweight Third Party Authentication). Um protocolo que utiliza criptografia para suportar a segurança em um ambiente distribuído.

Manipulador de exceção. Um conjunto de rotinas que respondem a uma condição anormal. Um manipulador de exceções é capaz de interromper e retomar a execução normal dos processos.

mapa.

1. Uma estrutura de dados que mapeia chaves em valores.

2. Um arquivo que define a transformação entre origens e destinos.

. No ambiente de desenvolvimento do EJB, a especificação de como os campos persistentes gerenciados por contêiner de um enterprise bean corresponde às colunas em uma tabela de banco de dados relacional ou outro armazenamento persistente.

máquina virtual. Uma especificação abstrata para um dispositivo de computação que pode ser implementada de formas diferentes no software e no hardware.

MBean. Consulte Managed Bean.

MBean (Managed Bean). Na especificação JMX (Java Management Extensions), os objetos Java que implementam os recursos e suas instrumentações.

método. Na programação orientada a objetos, uma operação que um objeto pode desempenhar. Um objeto pode ter muitos métodos.

Método create. Em beans corporativos, um método definido na interface inicial e chamado por um cliente para criar um enterprise bean. (Sun)

Método getter. Um método cujo objetivo é obter o valor de uma instância ou variável de classe. Isso permite que outro objeto descubra o valor de uma de suas variáveis.

Método setter. Um método cuja finalidade é definir o valor de uma instância ou variável de classe. Esse recurso permite que um outro objeto defina o valor de uma de suas variáveis.

metric. Um portador de informações, geralmente uma medida de desempenho de negócios, em um contexto de monitoramento.

modo de manutenção. Um estado de um nó ou servidor que um administrador pode utilizar para diagnosticar, manter ou ajustar o nó ou o servidor sem interromper o tráfego de entrada em um ambiente de produção.

modo silencioso. Um método para instalação ou desinstalação de um componente do produto a partir da linha de comandos sem exibição de GUI. Ao usar o modo silencioso, você especifica os dados necessários pelo programa de instalação ou desinstalação diretamente na linha de comandos ou em um arquivo (chamado um arquivo de opções ou arquivo de resposta).

Módulo EJB. Uma unidade de software que consiste em um ou mais beans corporativos e um descritor de desenvolvimento EJB. (Sun)

Navegador da Web. Um programa do cliente que inicia pedidos para um servidor Web e exibe as informações que o servidor retorna.

node.

1. Um agrupamento lógico de servidores gerenciados.

2. Qualquer item em um controle em árvore, incluindo um único elemento, elemento composto, comando de mapeamento, comentário ou nó de grupo.

3. Em XML, a menor unidade de uma estrutura válida e completa em um documento.

4. As formas fundamentais que compõem um diagrama.

nó-filho. Um nó dentro do escopo de outro nó.

Nome de Recurso Uniforme (URN). Um nome que identifica exclusivamente um serviço da Web para um cliente.

nome do host.

1. Na comunicação de Internet, o nome dado a um computador. O nome do host pode ser um nome completo de domínio como mycomputer.city.company.com, ou pode ser um subnome específico como mycomputer.

2. O nome da rede para um adaptador de rede em uma máquina física na qual o nó está instalado.

nome longo. A propriedade que especifica o nome lógico para o servidor na plataforma z/OS.

número da porta. Nas comunicações da Internet, o identificador de um conector lógico entre uma entidade do aplicativo e o serviço de transporte.

ObjectGrid. Um banco de dados de memória ativado por grade para aplicativos que são gravados em Java. ObjectGrid pode ser utilizado como um banco de dados em memória ou para distribuir dados pela rede.

objeto. Em design ou programação orientada por objeto, uma realização concreta (instância) de uma classe que consiste em dados e operações associadas aos dados. Um objeto contém os dados da ocorrência que são definidos pela classe, mas a classe pertence às operações que são associadas com os dados.

Objeto EJB. Nos beans corporativos, um objeto cuja classe implementa a interface remota do enterprise bean (Sun).

objeto genérico. Um objeto utilizado na API chama as expressões do XPATH para se referir a conceitos, entidades customizadas ou coletas. Por exemplo, a expressão do XPATH /WSRR/GenericObject recuperará todos os conceitos do WebSphere Service Registry and Repository.

objeto inicial de EJB. No EJB (Enteprise JavaBeans), um objeto que fornece as operações de ciclo de vida (criar, remover, localizar) de um enterprise bean. (Sun)

ODBC. Consulte Open Database Connectivity.

Open Database Connectivity (ODBC). Uma API (Interface de Programação de Aplicativo) padrão para acessar dados em sistemas de gerenciamento de bancos de dados relacionais e não-relacionais. Utilizando essa API, os aplicativos de bancos de dados podem acessar os dados armazenados em sistemas de gerenciamento de bancos de dados em vários computadores, mesmo se cada sistema de gerenciamento de bancos de dados utilizar um formato de armazenamento de dados e uma interface de programação diferente.

operação. Uma implementação de funções ou consultas para a qual um objeto pode ser chamado para desempenhar.

ORB. Consulte Object Request Broker.

ORB (Agente de Pedido de Objetos). Na programação orientada por objeto, o software que funciona como um intermediário, permitindo de maneira transparente que os objetos troquem pedidos e respostas.

organização. Uma entidade na qual as pessoas cooperam para realizar objetivos especificados, como uma corporação, uma empresa ou uma fábrica.

Pacote.

1. Em programação Java, um grupo de tipos. Os pacotes são declarados com a palavra-chave pacote. (Sun)

2. O wrapper em torno do conteúdo do documento que define o formato utilizado para transmitir um documento via Internet, por exemplo, RNIF, AS1 e AS2.

3. Para montar componentes em módulos e módulos em aplicativos corporativos.

pacote de instalação. Uma unidade instalável de um produto de software. Os pacotes de produto de software são unidades instaláveis separadamente que podem operar independente de outros pacotes desse produto de software.

Página JSP. Um documento baseado em texto que utiliza dados corrigidos do modelo e elementos JSP que descrevem como processar um pedido para criar uma resposta. (Sun)

painel. Uma página da Web que pode conter um ou mais visualizadores que representam graficamente os dados de negócio.

palavra-chave. Uma das palavras predefinidas de uma linguagem de programação, linguagem artificial, aplicativo ou comando.

pasta. Um contêiner utilizado para organizar objetos.

Perfil. Dados que descrevem as características de um usuário, grupo, recurso, programa, dispositivo ou local remoto.

permissão. Autorização para executar atividades, como ler e gravar arquivos locais, criar conexões de rede e carregar código nativo.

Persistência.

1. Uma característica de dados que é mantida pelos limites de sessões ou de um objeto que continua existindo após a execução do programa ou processo que o criou, geralmente em um armazenamento não-volátil como um sistema de banco de dados.

2. Em Java EE, o protocolo para transferir o estado de um bean de entidade entre suas variáveis de instância e um banco de dados subjacente. (Sun)

Persistir. Ser mantido além dos limites da sessão, geralmente em armazenamento não-volátil, como um sistema de banco de dados ou um diretório.

plano de construção. Um arquivo XML que define o processamento necessário para construir saídas de geração e que especifica a máquina em que o processamento ocorre.

Plataforma Java. Um termo coletivo para a linguagem Java para gravar programas; um conjunto de APIs, bibliotecas de classe e outros programas utilizados em programas de desenvolvimento, compilação e verificação de erros; e um Java virtual machine que carrega e executa arquivos de classe. (Sun)

Plataforma Java, Edição Padrão (Java SE). A plataforma da tecnologia Java principal. (Sun)

Plug-in do. Um módulo de software que pode ser instalado separadamente que inclui função a um programa, aplicativo ou interface existente.

Plug-in do servidor Web. Um módulo de software que suporta o servidor da Web em pedidos de comunicação de conteúdo dinâmico, como servlets, para o servidor de aplicativos.

PMI. Consulte Performance Monitoring Infrastructure.

PMI (Performance Monitoring Infrastructure). Um conjunto de pacotes e bibliotecas atribuídas para reunir, distribuir, processar e exibir dados de desempenho.

política. Um conjunto de considerações que influenciam o comportamento de um recurso gerenciado ou um usuário.

política de autorização. Uma política cujo destino é um serviço de negócios e cujo contrato contém uma ou mais asserções que concedem permissão para executar uma ação de canal.

política de implementação. Uma maneira opcional de configurar um ambiente eXtreme Scale baseado em vários itens, incluindo: número de sistemas, servidores, partições, réplicas (incluindo tipo de réplica), e tamanhos de heap para cada servidor.

política HA. Um conjunto de regras definido para um grupo HA que dita se zero (0) ou mais membros estão ativados. A política está associada a um grupo HA específico, correspondendo os critérios de correspondência de política com o nome do grupo.

Ponto-a-ponto. Um estilo de aplicativo de sistema de mensagens no qual o aplicativo de envio conhece o destino da mensagem.

Ponto de acesso de período do proxy. Um meio de identificar as configurações de comunicação para um ponto de acesso de período que não pode ser acessado diretamente.

ponto de interrupção. Um ponto marcado em um processo ou fluxo programático que faz com que esse fluxo seja pausado quando o ponto é atingido, geralmente para permitir a depuração ou a monitoração.

ponto de interrupção de entrada. Um ponto de interrupção configurado em um elemento do componente, que é selecionado antes do elemento do componente ser chamado.

porta listener. Um objeto que define a associação entre uma fábrica de conexões, um destino e um bean orientado por mensagem implementado. As portas de atendentes simplificam a administração das associações entre esses recursos.

processo.

1. Um procedimento progressivamente contínuo que consiste em uma série de atividades controladas que estão direcionadas sistematicamente para um determinado resultado ou fim.

2. A seqüência de documentos ou mensagens a serem trocadas entre Gerenciadores de Comunidade e participantes para executar uma transação de negócios.

processo síncrono. Um processo que é iniciado chamando uma operação de pedido-resposta. O resultado do processo é retornado pela mesma operação.

programação orientada a objetos. Uma abordagem de programação baseada nos conceitos de abstração e herança de dados. Diferente das técnicas de programação de procedimentos, a programação orientada a objetos concentra-se não em como algo é realizado, mas em quais objetos de dados consistem o problema e como eles são manipulados.

program temporary fix (PTF). Para produtos System i, System p e System z, uma correção testada pela IBM e disponibilizada para todos os clientes. Consulte também fix pack.

projeto do aplicativo corporativo (projeto EAR). Uma estrutura e hierarquia de pastas e arquivos que contém um descritor de desenvolvimento e documento de extensão IBM, além de arquivos que são comuns para todos os módulos Java EE que são definidos no descritor de desenvolvimento.

projeto EAR. Consulte projeto do aplicativo corporativo.

Projeto EJB. Um projeto que contém os recursos necessários para aplicativos EJB, incluindo beans corporativos; interfaces home, local e remota; arquivos JSP; servlets; e descritores de implementação.

Projeto Java. No Eclipse, um projeto que contém código fonte Java compilável e é um contêiner para pastas ou pacotes de origem.

prompt. Um componente de uma ação, que indica que a entrada do usuário é requerida para um campo antes da transição para uma tela de saída.

Propriedade. Uma característica de um objeto que descreve o objeto. Uma propriedade pode ser alterada ou modificada. As propriedades podem descrever um nome, tipo, valor ou comportamento de objeto, entre outras coisas.

Protocolo da Internet (IP). Um protocolo que roteia dados por meio de uma rede ou redes interconectadas. Este protocolo age como um intermediário entre as camadas mais altas do protocolo e a rede física.

Protocolo de Controle de Transmissões/Protocolo da Internet (TCP/IP). Um conjunto de protocolos de comunicação não-proprietário padrão de mercado que fornece conexões de ponta a ponta confiáveis entre aplicativos por redes interconectadas de tipos diferentes.

Provedor MBean. Uma biblioteca que contém uma implementação de um MBean JMX (Java Management Extensions) e seu arquivo descritor MBean XML (Extensible Markup Language).

proxy. Um gateway de aplicativo de uma rede para outra para um aplicativo de rede específico, como Telnet ou FTP por exemplo, em que um servidor proxy Telnet de firewall desempenha autenticação do usuário e, em seguida, permite que o tráfego flua pelo proxy como se não estivesse lá. A função é desempenhada no firewall e não na estação de trabalho do cliente, gerando maior carga no firewall.

PTF. Consulte correção temporária do programa .

public.

1. Na programação orientada a objetos, relativo a um membro de classe que é acessível a todas as classes.

2. Em linguagem de programação Java, relativo a um método ou variável que pode ser acessada por elementos que residem em outras classes. (Sun)

pulsação. Um sinal que uma entidade envia para outra para informar que ainda está ativa.

QoS. Consulte quality of service.

qualidade de serviço (QoS). Um conjunto de características de comunicação que um aplicativo precisa. A QoS (Quality of Service) define uma prioridade de transmissão específica, o nível de confiabilidade da rota e o nível de segurança.

qualificador. Um elemento simples que fornece um significado específico a outro elemento genérico, composto ou simples. Qualificadores são utilizados em ocorrências múltiplas ou única de mapeamento. Um qualificador pode ser utilizado também para denotar o espaço de nomes utilizado para interpretar a segunda parte do nome, geralmente referenciada como o ID.

rastreio de execução. Uma cadeia de eventos, registrados e exibidos em um formato hierárquico na página Eventos do cliente de teste de integração.

recebimento de dados. Relativo à direção do fluxo, que é do primeiro nó no processo (envio de dados) para o último nó no processo (recebimento de dados).

recursivo. Uma técnica de programação na qual um programa ou rotina chama a si mesmo para executar etapas sucessivas em uma operação, em que cada etapa utiliza a saída da etapa anterior.

recurso.

1. Um ativo discreto como, por exemplo, conjuntos de aplicativos, aplicativos, serviços de negócios, interfaces, terminais e eventos de negócio.

2. Um recurso de um sistema de cálculo ou sistema operacional necessário por um trabalho, tarefa ou programa em execução. Os recursos incluem armazenamento principal, dispositivos de entrada/saída, a unidade de processamento, conjuntos de dados, arquivos, bibliotecas, servidores de aplicativos e programas de controle ou processamento.

3. Uma pessoa, parte de equipamento ou material que é usado para execução de uma tarefa ou um projeto. Cada recurso é uma ocorrência ou exemplo em particular de uma definição de recurso.

Recurso da instância de cache. Um local onde qualquer aplicativo J2EE (Java 2 Platform, Enterprise Edition) pode armazenar, distribuir e compartilhar dados.

recurso de particionamento. Uma estrutura de programação e uma infra-estrutura de gerenciamento de sistemas que suporta o conceito de particionamento para enterprise bean, tráfego de HTTP e acesso ao banco de dados.

referência do EJB. Um nome lógico utilizado por um aplicativo para localizar a interface inicial de um enterprise bean no ambiente operacional de destino.

Refresh pack. Uma coleta acumulativa de correções que contêm novas funções. Consulte também fix pack, correção temporária.

Região. Uma área contígua de armazenamento virtual que possui características comuns e que pode ser compartilhada entre os processos.

Região servant. Uma área contígua de armazenamento virtual que é iniciada dinamicamente conforme a carga aumenta e é automaticamente parada conforme a carga se ameniza.

regra if-then. Uma regra na qual a ação (parte then) é executada somente quando a condição (parte if) é verdade.

Rendimento do processamento. A medida da quantidade de trabalho executado por um dispositivo, como um computador ou impressora, por um período e tempo, por exemplo, número de jobs por dia.

réplica. Um servidor que contém uma cópia do diretório ou diretórios de outro servidor. As réplicas fazem backup dos servidores para melhora o desempenho e os tempos de resposta para garantir a integridade dos dados.

réplica assíncrona. Um shard que recebe atualizações após os as confirmações de transação. Este método á mais rápido que uma réplica síncrona, mas tem a possibilidade de perda de dados porque a réplica assíncrona pode estar várias transações atrás do shard primário.

replicação. O processo de manutenção de um conjunto definido de dados em mais de um local. A replicação envolve a cópia de alterações designadas de um local (uma origem) para outro (um destino) e sincronização de dados nos dois locais.

Replicação de cache. O compartilhamento de IDs de cache, entradas de cache e invalidações de cache com outros servidores no mesmo domínio de replicação.

réplica síncrona. Um shard que recebe atualizações como parte da transação no shard primário para garantir a consistência de dados, o que pode aumentar o tempo de resposta comparado com uma réplica assíncrona.

Reprovado. Relativo a uma entidade, como um recurso ou um elemento de programação, que é suportada mas não mais recomendada e que pode tornar-se obsoleta.

restrição de cronometragem. Uma ação de validação especializada utilizada para medir a duração de uma chamada de método ou uma seqüência de chamadas de métodos.

root. O nome do usuário para o usuário do sistema com a autoridade maior.

script. Uma série de comandos, combinados em um arquivo, que executa uma função específica quando o arquivo é executado. Os scripts são interpretados conforme eles são executados.

Scripts. Um estilo de programação que reutiliza componentes existentes como base para criar aplicativos.

script shell. Um programa, ou script, que é interpretado pelo shell de um sistema operacional.

SDK. Consulte kit de desenvolvimento de software.

Segurança do Java Connector. Uma arquitetura projetada para estender o modelo de segurança de ponta a ponta para aplicativos com base em Java EE para incluir EIS (Enterprise Information Systems).

segurança global. Relativa a todos os aplicativos em execução no ambiente e determina se a segurança é utilizada, o tipo de registro utilizado para autenticação e outros valores, muitos dos quais atuam como padrões.

Separação do servidor Web. Uma topologia na qual o servidor Web é fisicamente separado do servidor de aplicativos.

Será impresso exatamente o que está na tela (WYSIWYG). Um recurso de um editor para exibir páginas continuamente exatamente como serão impressas ou apresentadas.

serialização. Em programação orientada a objetos, a gravação de dados de maneira seqüencial em uma mídia de comunicações a partir da memória do programa.

serializador. Um método de conversão de dados de objetos para outro formato, como binário ou XML.

server. Um programa de software ou um computador que fornece serviços a outros programas de software ou outros computadores. Consulte também host.

serviço de catálogo. Um serviço que controla o posicionamento dos shards e descobre e monitora o funcionamento de contêineres.

servidor da Web. Software capaz de atender a pedidos de HTTP (Hypertext Transfer Protocol).

servidor de aplicativos. Um programa do servidor em uma rede distribuída que fornece o ambiente de execução para um programa de aplicativo.

servidor de contêineres. Uma instância do servidor que pode hospedar vários shards. Uma Java virtual machine (JVM) pode hospedar vários servidores de contêineres.

servidor de monitoramento de TCP/IP. Um ambiente de tempo de execução que monitora todos os pedidos e respostas entre um navegador da Web e um servidor de aplicativos, bem como uma atividade de TCP/IP.

Servidor EJB. O software que fornece serviços a um contêiner EJB. Um servidor EJB pode hospedar um ou mais contêineres EJB. (Sun)

servidor independente. Um serviço de catálogo ou servidor de catálogo que é gerenciado a partir do sistema operacional que inicia e finaliza o processo do servidor.

servidor integrado. Um serviço de catálogo ou servidor de contêineres que reside em um processo existente e é iniciado e parado dentro do processo.

Servidor Java EE. Um ambiente de tempo de execução que fornece contêineres EJB ou de Web.

Servidor Proxy.

1. Um servidor que atua como um intermediário para as solicitações HTTP da Web que são hospedadas por um aplicativo ou um servidor da Web. Um servidor proxy atua como um substituto para os servidores de conteúdo na empresa.

2. Um servidor que recebe pedidos destinados a outro servidor e que atua em nome do cliente (como o proxy do cliente) para obter o serviço solicitado. Um servidor de proxy é freqüentemente utilizado quando o cliente e o servidor são incompatíveis para a conexão direta. Por exemplo, o cliente não consegue atender aos requisitos de autenticação de segurança do servidor, mas deve ter permissão para alguns serviços.

servlet. Um programa Java executado em um servidor da Web e que estende as funções do servidor através da geração de conteúdo dinâmico em resposta aos pedidos do Web client. Os servlets são utilizados geralmente para conectar os bancos de dados à Web.

sessão.

1. Uma conexão lógica ou virtual entre duas estações, programas de software ou dispositivos em uma rede que permite que os dois elementos se comuniquem e troquem dados.

2. Uma série de solicitações para um servlet de origem do mesmo usuário no mesmo navegador.

. No Java EE, um objeto usado por um servlet para controlar a interação de um usuário com um aplicativo da Web através de vários pedidos HTTP.

shard. Uma instância de uma partição. Um shard pode ser primário ou réplica.

sincronizar. Incluir, subtrair ou alterar um recurso ou artefato para corresponder ao outro.

Sintaxe. As regras para a construção de um comando ou instrução.

Sistema de mensagens assíncronas. Um método de comunicação entre programas em que um programa coloca uma mensagem em uma fila de mensagens e, em seguida, prossegue com seu próprio processamento sem esperar por uma resposta da sua mensagem.

Sistema de mensagens gerenciado por bean. Uma função de sistema de mensagens assíncronas que fornece a um enterprise bean controle completo sobre a infra-estrutura do sistema de mensagens.

sistema host. Um sistema de computador mainframe corporativo que hospeda aplicativos 3270. Nas ferramentas de desenvolvimento de terminal service 3270, o desenvolvedor usa o gravador do terminal service 3270 para conectar-se ao sistema host.

SLA. Consulte contrato de nível de serviço.

software livre. Software cujo código-fonte está publicamente disponível para uso ou modificação. O software livre geralmente é desenvolvido como uma colaboração pública e disponibilizado livremente, embora seu uso e sua redistribuição possam estar sujeitos a restrições de licenciamento. O Linux é um exemplo bem conhecido de software livre.

SQL. Consulte Linguagem de consulta estruturada.

SQL (Structured Query Language). Uma linguagem padronizada para definir e manipular dados em um banco de dados relacional.

SSL. Consulte Secure Sockets Layer.

SSL (Secure Sockets Layer). Um protocolo de segurança que fornece privacidade de comunicação. O SSL permite que aplicativos de cliente/servidor se comuniquem de forma designada para evitar escuta maliciosa, violação e falsificação de mensagens.

stack. Uma área na memória que geralmente armazena informações como registro temporário, valores de parâmetros e endereços de retorno de sub-rotinas e é baseada no princípio último a entrar, primeiro a sair (LIFO).

string. Em linguagens de programação, o formato dos dados utilizados para armazenar e manipular texto.

subclasse. Em Java, uma classe derivada de uma classe específica, por meio de herança.

subconsulta. Em SQL, uma subseleção utilizada em um predicado, por exemplo, uma instrução de seleção dentro da cláusula WHERE ou HAVING de outra instrução SQL.

suporte baseado em zona. Um função que ativa o posicionamento compartilhado baseado em regras para melhorar a disponibilidade da grade por meio da colocação de shards através de diferentes centros de dados, seja em andares diferentes ou mesmo em prédios ou geografias diferentes.

Tabela de autorizações. Uma tabela que contém a função para informações de mapeamento de usuário ou grupo que identificam o acesso permitido de um cliente a um recurso específico.

TCP. Consulte Protocolo de Controle de Transmissões.

TCP/IP. Consulte Protocolo de Controle de Transmissões/Protocolo da Internet.

TCP (Transmission Control Protocol). Um protocolo de comunicação utilizado na Internet e em qualquer rede que segue os padrões do IETF (Internet Engineering Task Force) para o protocolo de interligação de redes. O TCP oferece um protocolo confiável de host a host em redes de comunicação através da comutação de pacotes e em sistemas interconectados dessas redes.

tempo de ativação. O intervalo de tempo em segundos no qual uma entrada pode existir no cache antes de ser descartada.

tempo de compilação. O período de tempo durante o qual um programa de computador está sendo compilado em um programa executável.

Tempo de execução. O período de tempo durante o qual um programa de computador está em execução.

terminal.

1. Um aplicativo JCA ou outro consumidor cliente de um evento a partir do sistema de informações corporativos.

2. O sistema que é a origem ou o destino de uma sessão.

teste de componente. Um teste automatizado de um ou mais componentes de um aplicativo corporativo, que pode incluir classes Java, beans EJB ou serviços da Web.

timeout. Um intervalo de tempo concedido para que um evento ocorra ou seja concluído, antes da operação ser interrompida.

tipo primitivo. Em Java, uma categoria de tipo de dados que descreve uma variável que contém um valor único do tamanho apropriado e o formato para seu tipo: um número um caractere ou um valor Booleano. Exemplos de tipos primitivos incluem byte, short, int, long, float, double, char, boolean.

Tivoli Performance Viewer. Um cliente Java que recupera os dados PMI (Performance Monitoring Infrastructure) de um servidor de aplicativos e os exibe em diversos formatos.

token.

1. Um marcador usado para controlar o estado atual de uma instância de processo durante uma execução de simulação.

2. Uma mensagem específica ou padrão de bits que significa permissão ou controle temporário para transmitir através de uma rede.

Token de segurança. Uma representação de um conjunto de alegações que são feitas por um cliente que pode incluir um nome, senha, identidade, chave, certificado, grupo, privilégio e assim por diante.

topologia. O mapeamento físico ou lógico o local dos componentes de rede ou nós dentro de uma rede. As topologias de rede incluem barramento, anel, estrela e árvore.

topologia da implementação. A configuração de servidores e clusters em um ambiente de implementação e os relacionamentos físicos e lógicos entre eles.

topologia do tempo de execução. Uma descrição do estado momentâneo do ambiente.

transação. Um processo no qual todas as modificações de dados que foram feitas durante uma transação são confirmadas juntas como uma unidade ou revertidas como uma unidade.

transação global. Uma unidade de trabalho recuperável realizada por um ou mais gerenciadores de recursos em um ambiente de transação distribuído e coordenado por um gerenciador de transações externo.

type.

1. Em programação Java, uma classe ou interface.

2. Em um documento WSDL, um elemento que contém definições de tipo de dados usando algum sistema de tipo (como XSD).

UDDI. Consulte Descrição, Descoberta e Integração Universal.

Unidade de compilação. Uma parte de um programa de computador suficientemente completa para ser compilada corretamente.

Universally Unique Identifier (UUID). O identificador numérico de 128 bits utilizado para assegurar que dois componentes não têm o mesmo identificador.

UNIX System Services. Um elemento de z/OS que cria um ambiente UNIX de acordo com as especificações XPG4 UNIX 1995 e que fornece duas interfaces de sistema aberto no sistema operacional z/OS: uma API (interface de programação de aplicativos) e uma interface shell interativa.

URI. Consulte Identificador Uniforme de Recursos (URI).

URI (Uniform Resource Identifier).

1. Uma cadeia compacta de caracteres para identificação de um recurso abstrato ou físico.

2. Um endereço exclusivo que é usado para identificar o conteúdo na Web, como uma página de texto, um vídeo ou clipe de som, uma imagem parada ou animada, ou um programa. A forma mais comum de URI é o endereço de páginas da Web, que é uma forma particular ou um subconjunto de URIs chamado URL (Uniform Resource Locator). Normalmente, um URI descreve como acessar o recurso, o computador que contém o recurso e o nome do recurso (o nome de um arquivo) no computador.

URL. Consulte Localizador Uniforme de Recursos.

URL (Uniform Resource Locator). O endereço exclusivo de um recurso de informações que pode ser acessado em uma rede como a Internet. A URL inclui o nome abreviado do protocolo utilizado para acessar o recurso de informações e as informações utilizadas pelo protocolo para localizar o recurso de informações.

URN. Consulte Nome Uniforme de Recursos.

usuário autenticado. Um usuário do portal que não efetuou login com uma conta válida (ID do usuário e senha). Os usuários autenticados têm acesso a todos os locais públicos.

utilitário de carga. Um componente que lê dados de um armazenamento persistente e grava dados nele.

UUID. Consulte Identificador Universalmente Exclusivo.

Variável. Uma representação de um valor alterável.

Variável de ambiente. Uma variável que especifica como um sistema operacional ou outro programa é executado ou os dispositivos que o sistema operacional reconhece.

variável global. Uma variável que é utilizada para manter e manipular valores designados a ela durante a conversão e que é compartilhada por meio de mapas e conversões de documentos. Um dos três tipos de variáveis com suporte na linguagem de comandos de mapeamento do Data Interchange Services.

versão. Um programa licenciado separadamente que geralmente possui um novo código ou uma nova função significativa.

virtualização. Uma técnica que encapsula as características de recursos da maneira na qual os outros sistemas interagem com esses recursos.

WAR. Consulte Archive da Web.

WAR (Web Archive). Um formato de arquivo compactado, definido pelo padrão Java EE, para armazenar todos os recursos necessários para instalar e executar um aplicativo da Web em um único arquivo. Consulte também archive corporativo.

WCCM. Consulte WebSphere Common Configuration Model.

Web site. Uma coleta de arquivos relacionados disponível na Web que é gerenciada por uma única entidade (uma organização ou uma pessoa) e contém informações em hipertexto para seus usuários. Um Web site geralmente inclui links de hipertexto para outros Web sites.

WebSphere. Um nome de marca IBM que abrange ferramentas para desenvolvimento de aplicativos de e-business e middleware para execução de aplicativos da Web.

WebSphere Common Configuration Model (WCCM). Um modelo que fornece acesso programático aos dados de configuração.

WLM. Consulte Workload Manager.

Workload Manager (WLM). Um componente do z/OS que fornece a capacidade de executar várias cargas de trabalho ao mesmo tempo dentro de uma imagem ou de várias imagens do z/OS.

WYSIWYG. Consulte Será impresso exatamente o que está na tela .

XA. Uma interface bidirecional entre um ou mais gerenciadores de recursos que fornecem acesso aos recursos compartilhados e um gerenciador de transações que monitora e resolve as transações.

XML. Consulte Linguagem de Marcação Extensível.

X/Open XA. A interface X/Open Distributed Transaction Processing XA. Um padrão proposto para comunicação de transação distribuída. O padrão especifica uma interface bidirecional entre os gerenciadores de recursos que fornecem acesso a recursos compartilhados dentro de transações e entre um serviço de transação que monitora e resolve as transações.

z/OS. Um sistema operacional mainframe IBM que usa armazenamento real de 64 bits.

Avisos

Referências nesta publicação a produtos, programas ou serviços IBM não significam que a IBM pretende torná-los disponíveis em todos os países onde opera. Qualquer referência a um produto, programa ou serviço IBM não significa que apenas um produto, programa ou serviço IBM possa ser utilizado. Qualquer produto, programa ou serviço funcionalmente equivalente, que não infrinja nenhum direito de propriedade intelectual da IBM ou outros direitos legalmente protegidos, poderá ser utilizado em substituição a este produto, programa ou serviço. A avaliação e a verificação da operação em conjunto com outros produtos, exceto aqueles expressamente designados pela IBM, são de inteira responsabilidade do usuário.

A IBM pode ter patentes ou solicitações de patentes pendentes relativas a assuntos tratados nesta publicação. O fornecimento desta publicação não garante ao Cliente nenhum direito sobre tais patentes. Pedidos de licença devem ser enviados, por escrito, para:

Gerência de Relações Comerciais e Industriais da IBM Brasil
IBM Corporation
Rio de Janeiro, RJ
CEP 22290-240

Licenciados deste programa que desejam obter mais informações sobre este assunto com objetivo de permitir: (i) a troca de informações entre programas criados independentemente e outros programas (incluindo este) e (ii) a utilização mútua das informações trocadas, devem entrar em contato com:

IBM Corporation
Av. Pasteur, 138-146, Botafogo
Botafogo
Rio de Janeiro, RJ
CEP 22290-240

Tais informações podem estar disponíveis, sujeitas a termos e condições apropriados, incluindo em alguns casos, o pagamento de uma taxa.

Marcas Registradas

Os termos a seguir são marcas registradas da IBM Corporation nos Estados Unidos e/ou em outros países:

- AIX
- CICS
- Cloudscape
- DB2
- Domino
- IBM
- Lotus
- RACF
- Redbooks
- Tivoli
- WebSphere
- z/OS

Java e todas as marcas registradas baseadas em Java são marcas registradas da Sun Microsystems, Inc. nos Estados Unidos e/ou em outros países.

LINUX é uma marca registrada de Linus Torvalds nos Estados Unidos e/ou em outros países.

Microsoft, Windows, Windows NT e o logotipo Windows são marcas registradas da Microsoft Corporation nos Estados Unidos e/ou em outros países.

UNIX é uma marca registrada do The Open Group nos Estados Unidos e em outros países.

Outros nomes de empresas, produtos e serviços podem ser marcas registradas ou marcas de serviço de terceiros.

Índice Remissivo

A

administrando 215
  WebSphere Application Server 230
ajuste 8, 9, 10, 11
API de estatísticas 247
armazenamento em cache 119
arquivo de definição de construção
  criando
    CIP 33
    IIP 36
arquivo de resposta 50
Arquivo objectGrid.xsd 166
Arquivo objectGridSecurity.xsd 187
arquivo orb.properties 30
arquivos de extensão 55
atualizações falhas 120
atualizador de dados baseado em tempo 96
autônomos 53, 218
autorização de cliente
  acesso apenas do criador 290
  customizado 290
  JAAS 290
  permissões
    período de verificação 290
autorização de grade 295

B

backend 120
bean gerenciado 265
Beans de Extensão Sprint 202
Beans Gerenciados 265
benefícios 114
bloqueio
  configurando com XML 69
  configurando programaticamente 69
  não 69
  otimista 69
  pessimista 69
boas práticas 212, 310

C

CA Wily Introscope 274
cliente 72
Comando manageprofiles 40, 43
Comando wasprofile 40, 42
configuração 187
  implementação
    distribuído 7
    local 7
configurando 53, 59, 142
Configurando 72
configurando após a instalação 41
Console de Primeiras Etapas 41
contêineres 15

D

definições de customização
  gerando 57
desempenho 310
desinstalando 54
dimensionando CPU 24, 25
disponibilidade
  definindo 215
distribuindo alterações
  JVMs peer 136

E

elemento de log 136
estado de sessão
  J2EE 244
  WebSphere Application Server Community Edition 244
estatísticas 249
estratégia de colocação 21
evictors
  configurando 125
  Evictor TTL 125
  plug-in 130

F

failover
  configurando 13
Ferramenta de Gerenciamento de Perfis 55, 57

G

gerenciador de sessões 237
gerenciador de sessões HTTP 235
  com o WebSphere Application Server Community Edition 242
  com WebSphere Virtual Enterprise 237
  parâmetros para configuração 240
grade 21

H

Hyperic HQ 277

I

IBM Installation Factory
  arquivo de definição de construção 32
IBM Tivoli Monitoring 268
IBM Update Installer para WebSphere
  desinstalando
    CIP 36
IBM Update Installer para WebSphere Software 52
índice
  configuração 131
  HashIndex 131
iniciando servidores 218
instalando 53
  autônomos 28
  IBM Installation Factory
    CIP 32
    IIP 32
  manutenção 52
  Network Deployment 30
  pacote de instalação customizada 39
  server 28
  silenciosamente 39, 50, 51
  WebSphere Application Server 30
Installation Factory
  CIP
    manutenção 35
interface ObjectGrid 60
Interface ObjectGridManager
  ativando rastreio com 279, 305
Introscope 274
invalidação 139
invalidação do cliente 77, 134

J

Java Authentication and Authorization Service
  JAAS 287
Java Persistence API 96
Java Persistence API (JPA) 93
  plug-in de cache
    introdução 97
  topologia de cache
    integrado 97
    particionado integrado 97
    remoto 97
Java virtual machine 11
JMS 139
JPA (Java Persistence API)
  plug-in de cache
    configuração 100
  plug-in Hibernate
    configuração 103
  plug-in OpenJPA
    configuração 109
  topologia de cache
    Hibernate 103
    integrado 100, 103, 109
    OpenJPA 109
    particionado integrado 100, 103, 109
    remoto 100, 103, 109
JVM 11

L

linha de comandos 51
lista de verificação operacional 18

listener de evento 139
logs
  visão geral 279, 305

M

mapa 60
mapa de apoio
  estratégia de bloqueio 67
  plug-ins 63
  Sessão 63
MBean 265
mensagens 309
metadados de entidade
  Arquivo emd.xsd 182
  Configuração XML 171, 182
métricas 268, 277
migrar 28
módulos estatísticos 251
monitor 277
monitoramento 253
  agente 268
  com a API de estatísticas 249
  com ferramentas do fornecedor 268
Monitorando Aplicativos
  com o Introscope 274

N

Network Deployment 43
notas sobre o release 309

O

object request broker 198
Object Request Broker 10, 30, 53
ObjectGrid
  Configuração XML 166
ORB 10
orb.properties 198

P

parâmetros 50, 51
parando servidores 228
partição 21
peer 133
Perfil
  alterando 42
  aumentando 40
  criando 40, 41
perfis
  alteração 43
  criar 43
  usuário não-root 49
performance monitoring
  infrastructure 253, 257
Performance Monitoring
  Infrastructure 253
personalização 55
planejamento de capacidade 21
planejando 7, 8, 9, 18
plug-in 60
plug-in do Installation Factory
  arquivo de definição de construção
    modificar 38
  instalando
    CIP 34
    IIP 37
plug-in Profile Management Tool 40, 41, 42
PMI i, 257
  Veja também performance monitoring infrastructure
  MBean 247
PMI (Performance Monitoring Infrastructure) 247
política de implementação 142
  Arquivo deploymentPolicy.xsd 148
  Configuração XML 148
  XML descritor 145
por partição 24
portas de rede 8
processo do serviço de catálogo
  iniciando no WebSphere Application Server 230
processos do contêiner
  iniciando 221
propriedades 188
  cliente 195
  servidor 189
propriedades do cliente 195
propriedades do servidor 189

R

rede 9
replicação 133, 139
resolução de problemas 305
  mensagens 309
  notas sobre o release 309

S

segurança
  autenticação
    criando um autenticador 287
    LDAP 287
    Tivoli access manager 287
    WebSphere Application Server 287
  Conexão Única (SSO) 287
  Configuração XML 184, 298
  Credencial 287
  integração 285
  introdução 285
  Local 287
  plug-ins 287
segurança. 187, 302
  integração 301
  WebSphere Application Server 301
segurança de grade
  gerenciador de tokens 286
  JSSE 286
segurança do cliente-servidor
  secure sockets layer (SSL) 293
  TCP/IP 293
  transport layer security (TLS) 293
segurança JMXControle de acesso
  autenticação 296
  suporte JAAS 296
  transporte seguro 296
sequência de log 136
serviço de catálogo
  iniciando
    em um ambiente que não está executando o WebSphere Application Server 218
    em um WebSphere Application Server Environment 218
servidor de catálogos
  armazenamento em cluster 15
  ativando logs 279, 305
  ativando rastreio 279, 305
  iniciando 215
  Parando 215
servidor de contêineres
  ativando logs 279, 305
  ativando rastreio 279, 305
  iniciando 215, 223
  Parando 215
Sessão 60
sessões 235
shard 21
silenciosamente 54
SIP
  gerenciamento de sessões 241
  sessão 241
sistema de mensagens Java 133
sistemas operacionais 9
Spring
  arquivo objectgrid.xsd 211
  beans de extensão 200
  compactando 200
  Configuração XML 211
  escopo do shard 200
  estrutura 200
  fluxo da Web 200
  suporte a espaço de nomes 200
  transações nativas 200
  XML descritor 206
startOgserver
  iniciando 223
stopOgserver 229
suporte 114, 309
suporte a armazenamento em cache 114
suporte a armazenamento em cacheutilitário de cargatransação do utilitário de carga 114

T

tarefas 55
tarefas customizadas
  em execução 58
  fazendo upload 58
tempo de resposta 212
tempo real 212
Tivoli 268
trace
  opções para configuração 282, 307
  visão geral 279, 305
Transação 60
  ID 88
  retorno de chamada 88

transação do utilitário de carga 120
transações paralelas 25

U

utilitário de amostra xsadmin 266
utilitário de carga 120
  pré-carregamento 88
utilitários de carga
  JPA 93

V

visão geralprogramaticamente
  utilizando um utilitário de carga 86

W

WebSphere Application Server 42, 43, 52
WebSphere Configuration Tools 55
WebSphere Customization Tools 55, 57
Wily 274
Wily Introscope 274
write-behind 114, 119, 120
  atualizações falhas
    manipulação 121

X

XML 142
XML Descritor do ObjectGrid 150

Impresso em Brazil