Plataforma do Aplicativo JBoss Enterprise 5.0

Guia de Inicialização

para Uso com a Plataforma do Aplicativo do JBoss Enterprise 5.0

Red Hat Grupo de Documentação

Plataforma do Aplicativo JBoss Enterprise 5.0 Guia de Inicialização
para Uso com a Plataforma do Aplicativo do JBoss Enterprise 5.0
Edição 1.0

Autor Red Hat Grupo de Documentação

Este Guia de Inicialização descreve informação relevante sobre o uso inicial da Plataforma do Aplicativo do JBoss Enterprise

Introdução

A Plataforma do Aplicativo JBoss Enterprise é uma implementação de código aberto do Java EE suite de serviços. Ela engloba um conjunto de ofertas para os clientes empresariais que buscam pelos perfis pré-configurados dos componentes do JBoss Enterprise Middleware que foram testados e certificados juntos para fornecerem uma experiência integrada. Uma arquitetura de servidor de uso fácil e de alta flexibilidade faz do JBoss a escolha ideal para os usuários começando com o J2EE, assim como arquitetos mais experientes que procuram por uma plataforma middleware personalizada.

Uma vez que ela é baseada no Java, a Plataforma do Aplicativo JBoss Enterprise é uma plataforma fácil de instalar e usar em qualquer sistema operacional que suporta o Java. O código fonte disponível de leitura é uma ferramenta potente de aprendizado para depurar e entender o servidor. Ela também fornece a flexibilidade de criar versões personalizadas para uso pessoal ou comercial.

1. Contribuição

Caso você encontre um erro tipográfico ou se você pensou em algo que possa melhorar este manual, submeta um relatório no in JIRA: *http://jira.jboss.com* em relação à *Plataforma do Aplicativo JBoss Enterprise* de produto e *Documentação* do componente.

Caso você tenha uma sugestão para aperfeiçoamento desta documentação, tente ser o mais específico possível em sua descrição. Caso você tenha encontrado um erro, por favor inclua o número de seção e alguns pedaços do texto em volta para que possamos localizar isto com maior facilidade.

O servidor JBoss - Uma passagem rápida

1.1. Estrutura do Servidor

Consulte o capítulo de Migração do Guia de Instalação que acompanha esta liberação da Plataforma do Aplicativo JBoss Enterprise, para um melhor entendimento da estrutura do servidor do aplicativo.

1.2. Iniciando e Interrompendo o Servidor

1.2.1. Iniciar o Servidor

Dirija-se ao diretório **`JBOSS_DIST/jboss-as/bin`** e execute os scripts **`run.bat`** (Windows) ou **`run.sh`** (Linux), de acordo com seu sistema operacional.

Observe que não há mensagem de **`Server Started`** exibida no console quando o servidor é iniciado usando o perfil **`production`**. Esta mensagem pode ser visualizada no arquivo **`server.log`** localizada no subdiretório **`server/production/log`**.

Conexão remota para o servidor da Plataforma do Aplicativo JBoss Enterprise

A Plataforma do Aplicativo do JBoss Enterprise vincula os próprios serviços para o localhost (127.0.0.1) por padrão, ao invés de vincular todas as interfaces disponíveis (0.0.0.0). Isto foi feito primariamente por razões de segurança, devido às preocupações de segurança dos usuários indo à produção sem terem garantido a segurança de seus servidores corretamente. Para ativar o acesso remoto pela vinculação dos serviços do JBoss a uma interface particular, apenas rode o jboss com a opção **`-b`**. Para vincular todas as interfaces disponíveis e re-ativar o comportamento de legacia, use o **`./run.sh -b 0.0.0.0`** no Linux e **`run.bat -b 0.0.0.0`** no Windows. Neste caso, conscientize-se que você precisa aplicar a segurança ao seu servidor de maneira adequada.

Usando o **`-b`** como parte da linha de comando do Servidor JBoss é equivalente à configuração destas propriedades individuais: -Djboss.bind.address, -Djava.rmi.server.hostname, -Djgroups.bind_addr e -Dbind.address. Passando -Djboss.bind.address para o processo Java como parte da variável JAVA_OPTS nos scripts de rodagem, não funcionará uma vez que isto é uma propriedade JBoss e não uma propriedade JVM.

Para maiores informações incluindo a configuração múltiplas das instâncias do servidor do JBoss numa única máquina e domínios múltiplos de host com o JBoss, por favor refira-se ao *Guia de Administração e Configuração*[1].

Na inicialização do seu servidor, o resultado de sua tela deve parecer-se com o abaixo (conta para diferentes diretórios de instalação) e não deve conter erro ou mensagens de exceção:

[1] http://www.redhat.com/docs/en-US/JBoss_Enterprise_Application_Platform/5.0.0/html-single/Administration_And_Configuration_Guide/index.html

```
[user@mypc bin]$ ./run.sh
=========================================================================

  JBoss Bootstrap Environment

  JBOSS_HOME: /home/user/jboss-as-version/jboss-as

  JAVA: java

  JAVA_OPTS: -Dprogram.name=run.sh -server -Xms1503m -Xmx1503m -Dsun.rmi.dgc.client.
gcInterval=3600000 -Dsun.rmi.dgc.server.gcInterval=3600000 -Djava.net.preferIPv4Stack=true

  CLASSPATH: /home/user/jboss-as-version/jboss-as/bin/run.jar

=========================================================================
```

Maiores opções do script **run** da Plataforma do Aplicativo JBoss Enterprise, estão discutidas na *Seção 1.2.2, "Iniciar o Servidor com Configuração Alternada"* abaixo.

Nota

Observe que não há mensagem de *Servidor Iniciado* exibida no console quando o servidor é iniciado usando o perfil **production**, que é o perfil padrão usado quando nenhum outro é especificado. Esta mensagem pode ser visualizada no arquivo **server.log** localizado no subdiretório **server/production/log**.

1.2.2. Iniciar o Servidor com Configuração Alternada

O uso de um **run.sh** sem qualquer argumento, inicia o servidor usando o conjunto de arquivo de configuração do servidor **default**. Para iniciar com um conjunto de arquivo de configuração alternado: passe o nome do conjunto de arquivo de configuração do servidor (o mesmo que o nome do diretório de configuração sob o **JBOSS_DIST/jboss-as/server**) que você deseja usar como valor para a opção de linha de comando **-c**. Por exemplo, para iniciar com o conjunto de arquivo de configuração **minimal** você deve especificar:

```
[bin]$ ./run.sh -c minimal
...
...
...
15:05:40,301 INFO  [Server] JBoss (MX MicroKernel) [5.0.0 (build: SVNTag=JBoss_5_0_0
 date=200801092200)] Started in 5s:75ms
```

1.2.3. Usando o run.sh

O script **run** suporta as seguintes opções:

```
usage: run.sh [options]
-h, --help                          Show help message
-V, --version                       Show version information
--                                  Stop processing options
-D<name>[=<value>]      Set a system property
-d, --bootdir=<dir>           Set the boot patch directory; Must be absolute or url
-p, --patchdir=<dir>          Set the patch directory; Must be absolute or url
-n, --netboot=<url>           Boot from net with the given url as base
-c, --configuration=<name>    Set the server configuration name
```

```
-B, --bootlib=<filename>      Add an extra library to the front bootclasspath
-L, --library=<filename>      Add an extra library to the loaders classpath
-C, --classpath=<url>         Add an extra url to the loaders classpath
-P, --properties=<url>        Load system properties from the given url
-b, --host=<host or ip>       Bind address for all JBoss services.
-g, --partition=<name>        HA Partition name (default=DefaultDomain)
-m, --mcast_port=<ip>         UDP multicast port; only used by JGroups
-u, --udp=<ip>                UDP multicast address
-l, --log=<log4j|jdk>         Specify the logger plugin type
```

1.2.4. Interrompendo o Servidor

Para desligar o servidor, simplesmente digite **`Ctrl-C`** no console pelo qual o JBoss foi iniciado. Como forma alternativa, você pode usar o comando **`shutdown.sh`**.

```
[bin]$ ./shutdown.sh -S
```

O script **`shutdown`** suporta as seguintes opções:

```
usage: shutdown [options] <operation>

options:
-h, --help                      Show this help message (default)
-D<name>[=<value>]  Set a system property
--                              Stop processing options
-s, --server=<url>        Specify the JNDI URL of the remote server
-n, --serverName=<url>    Specify the JMX name of the ServerImpl
-a, --adapter=<name>      Specify JNDI name of the MBeanServerConnection to use
-u, --user=<name>         Specify the username for authentication
-p, --password=<name>     Specify the password for authentication

operations:
-S, --shutdown                  Shutdown the server
-e, --exit=<code>         Force the VM to exit with a status code
-H, --halt=<code>         Force the VM to halt with a status code
```

O uso do comando shutdown, requer uma configuração de servidor que contenha o serviço **`jmx-invoker-service.xml`**. Portanto, você não pode usar o comando shutdown com a configuração **`minimal`**.

1.2.5. Executando como um Serviço sob o Microsoft Windows

A Plataforma do Aplicativo JBoss Enterprise 5 vem com os arquivos necessários para configurar o servidor a ser rodado como um serviço sob o windows. Distribuído com o JBoss Nativo, ele ativa a Plataforma do JBoss Enterprise 5 a ser rodada como um serviço do sistema operacional do Windows. Para instalar o serviço, navegue no **`JBOSS_DIST/jboss-as/bin`** e localize o arquivo **`service.bat`**. Rode o seguinte comando num aviso de comando:

```
./service.bat install
```

Este comando instala a Plataforma do Aplicativo JBoss Enterprise como um serviço. Sob a lista dos serviços do Windows você encontrará isto listado pelo nome abreviado **`JBAS50SVC`** e nome **`JBoss Application Server 5.1`**.

Uma vez que o serviço seja instalado com sucesso, você poderá controlar e configurar o serviço a partir do aplicativo **`Windows Services Manager`**. Você pode configurá-lo quando o sistema é

inicializado. Além disso, você pode iniciar e interromper o serviço a partir do **`Windows Services Manager`**.

Caso você deseje passar parâmetros ao servidor (por exemplo, **`-b`**, **`-c`**) quando rodando isto como um serviço, você pode realizá-lo pela edição do **`service.bat`** para incluir estes parâmetros:

```
call run.bat -c default -b localhost < .r.lock >> run.log 2>&1
```

Importante

Adicione estes parâmetros em ambas localizações onde o **`run.bat`** aparece neste arquivo **`service.bat`**.

1.3. O Console JMX

Quando o Servidor JBoss estiver rodando, você pode obter uma visualização do servidor através do aplicativo do console JMX no site *http://localhost:8080/jmx-console*.

Por padrão, o console JMX é assegurado e nem mesmo um usuário admin pode acessá-lo. Caso você deseje permitir acesso ao console JMX, vá ao diretório **`JBOSS_DIST/jboss-as/server/<instance-name>/conf/props/`** e descomente o admin `userid` e código `password` sobre o arquivo **`jmx-console-users.properties`**.

Isto permitirá que o usuário admin acesse o console JMX usando combinação de nome de usuário e senha especificada com o arquivo **`jmx-console-users.properties`**.

Importante

Caso você tenha alterado o arquivo **`jmx-console-users.properties`** quando o servidor estiver rodando, você terá que restaurar o servidor para as alterações fazerem efeito.

O Console JMX é o Console de Gerenciamento do JBoss que oferece uma visão bruta do JMX MBeans, a qual substitui o servidor. Eles podem oferecer diversas informações sobre a execução do servidor e permitir que você modifique sua configuração, inicie e interrompa os componentes e assim por diante.

Por exemplo, encontre o link **`service=JNDIView`** e clique nele. Este MBean em específico, fornece um serviço que permite que você visualize a estrutura dos espaços de nomes do JNDI dentro do servidor. Agora, encontre uma operação chamada **`list`** perto do botão da página de visualização do MBean e clique no botão **`invoke`**. A operação retorna a tela dos nomes atuais vinculados à árvore JNDI, o que é muito útil ao iniciar a implementação de seus próprios aplicativos e quiser saber porque você não consegue resolver um nome EJB em particular.

Observe alguns outros MBeans e suas operações listadas; tente mudar alguns recursos de configuração e veja o que acontece. Salvo algumas poucas exceções, nenhuma das mudanças realizadas através do console são persistentes. A configuração original será recarregada quando você reiniciar o JBoss, permitindo que você experimente livremente sem fazer nenhuma mudança permanente.

Nota

Se você instalou o JBoss usando o instalador gráfico, o Console JMX solicitará um nome de usuário e senha antes que você possa acessá-lo. Se você instalou usando outros módulos, você pode ainda configurar o JMX Security manualmente. Mostraremos como assegurar seu console no **`JBOSS_DIST/jboss-as/server/<instance-name>/conf/props/`**.

1.4. Serviço JNDIView

O Serviço JNDIView é ativado por padrão na Plataforma de Aplicativo JBoss Enterprise. Este serviço é listado no **`jmx-console`** (*http://localhost:8080/jmx-console*). Navegue ao **`jboss:service=JNDIView`** Mbean e clique naquele link. Você encontrará o método `list` na página de operações. Clique no botão adjacente **Invoke** para este método `list`.

A operação listar exibirá os conteúdos da árvore JNDI. O resultado observará algo similar com:

```
java: Namespace

+- securityManagement (class: org.jboss.security.integration.JNDIBasedSecurityManagement)
+- comp (class: javax.namingMain.Context)
+- XAConnectionFactory (class: org.jboss.jms.client.JBossConnectionFactory)
+- JmsXA (class: org.jboss.resource.adapter.jms.JmsConnectionFactoryImpl)
+- policyRegistration (class: org.jboss.security.plugins.JBossPolicyRegistration)
+- TransactionPropagationContextImporter (class:
 com.arjuna.ats.internal.jbossatx.jta.PropagationContextManager)
+- app (class: org.jnp.interfaces.NamingContext)
|   +- Manager (class: javax.inject.manager.Manager)
+- ClusteredConnectionFactory (class: org.jboss.jms.client.JBossConnectionFactory)
+- Mail (class: javax.mail.Session)
+- TransactionPropagationContextExporter (class:
 com.arjuna.ats.internal.jbossatx.jta.PropagationContextManager)
+- ProfileService (class:
 org.jboss.system.server.profileservice.repository.AbstractProfileService)
+- DefaultDS (class: org.jboss.resource.adapter.jdbc.WrapperDataSource)
+- jaas (class: javax.naming.Context)
|   +- HsqlDbRealm (class: org.jboss.security.plugins.SecurityDomainContext)
+- ClusteredXAConnectionFactory (class: org.jboss.jms.client.JBossConnectionFactory)
+- TransactionSynchronizationRegistry (class:

 com.arjuna.ats.internal.jta.transaction.arjunacore.TransactionSynchronizationRegistryImple)
+- SecurityProxyFactory (class: org.jboss.security.SubjectSecurityProxyFactory)
+- ConnectionFactory (class: org.jboss.jms.client.JBossConnectionFactory)
+- DefaultJMSProvider (class: org.jboss.jms.jndi.JNDIProviderAdapter)
+- TransactionManager (class: com.arjuna.ats.jbossatx.jta.TransactionManagerDelegate)
+- timedCacheFactory (class: javax.naming.Context)
Failed to lookup: timedCacheFactory, errmsg=org.jboss.util.TimedCachePolicy cannot be cast to
 javax.naming.NamingEnumeration


Global JNDI Namespace

+- UserTransactionSessionFactory (proxy: $Proxy109 implements interface
 org.jboss.tm.usertx.interfaces.UserTransactionSessionFactory)
+- UUIDKeyGeneratorFactory (class:
 org.jboss.ejb.plugins.keygenerator.uuid.UUIDKeyGeneratorFactory)
+- HiLoKeyGeneratorFactory (class:
 org.jboss.ejb.plugins.keygenerator.hilo.HiLoKeyGeneratorFactory)
+- SecureDeploymentManager (class: org.jnp.interfaces.NamingContext)
|   +- remote[link -> DeploymentManager] (class: javax.naming.LinkRef)
+- SecureManagementView (class: org.jnp.interfaces.NamingContext)
|   +- remote[link -> ManagementView] (class: javax.naming.LinkRef)
```

```
+- persistence.unit:unitName=jsfejb3.ear (class: org.jnp.interfaces.NamingContext)
|   +- app.jar#helloworld (class: org.hibernate.impl.SessionFactoryImpl)
+- DeploymentManager (class: org.jboss.aop.generatedproxies.AOPProxy$4)
+- XAConnectionFactory (class: org.jboss.jms.client.JBossConnectionFactory)
+- topic (class: org.jnp.interfaces.NamingContext)
+- ClusteredConnectionFactory (class: org.jboss.jms.client.JBossConnectionFactory)
+- ProfileService (class: org.jboss.aop.generatedproxies.AOPProxy$2)
+- SecureProfileService (class: org.jnp.interfaces.NamingContext)
|   +- remote[link -> ProfileService] (class: javax.naming.LinkRef)
+- queue (class: org.jnp.interfaces.NamingContext)
|   +- DLQ (class: org.jboss.jms.destination.JBossQueue)
|   +- ExpiryQueue (class: org.jboss.jms.destination.JBossQueue)
+- ClusteredXAConnectionFactory (class: org.jboss.jms.client.JBossConnectionFactory)
+- UserTransaction (class: org.jboss.tm.usertx.client.ClientUserTransaction)
+- ConnectionFactory (class: org.jboss.jms.client.JBossConnectionFactory)
+- jmx (class: org.jnp.interfaces.NamingContext)
|   +- invoker (class: org.jnp.interfaces.NamingContext)
|   |   +- RMIAdaptor (proxy: $Proxy103 implements interface
 org.jboss.jmx.adaptor.rmi.RMIAdaptor,interface org.jboss.jmx.adaptor.rmi.RMIAdaptorExt)
|   +- rmi (class: org.jnp.interfaces.NamingContext)
|   |   +- RMIAdaptor[link -> jmx/invoker/RMIAdaptor] (class: javax.naming.LinkRef)
+- TomcatAuthenticators (class: java.util.Properties)
+- console (class: org.jnp.interfaces.NamingContext)
|   +- PluginManager (proxy: $Proxy104 implements interface
 org.jboss.console.manager.PluginManagerMBean)
+- ManagementView (class: org.jboss.aop.generatedproxies.AOPProxy$3)
```

Isto detalha os nomes JNDI dos quais seus EJBs estão vinculados.

1.5. Hot deployment de serviços no JBoss.

Os serviços hot-deployable são aqueles que podem ser adicionados ou removidos do servidor em execução. Estes são colocados no diretório **`JBOSS_DIST/jboss-as/server/<instance-name>/deploy`**. Vamos dar uma olhada no exemplo prático do hot-deployment dos serviços no JBoss.

Inicie o JBoss, caso não esteja em execução, e observe o diretório **`server/production/deploy`**. Remova o arquivo **`mail-service.xml`** e veja o resultado do servidor:

```
13:10:05,235 INFO  [MailService] Mail service 'java:/Mail' removed from JNDI
```

Em seguida, substitua o arquivo e acompanhe o JBoss na re-instalação do serviço.

```
13:58:54,331 INFO  [MailService] Mail Service bound to java:/Mail
```

Isto é um hot deployment em ação.

1.5.1. Configurações Hot deployment

O Hot deployment de serviços no servidor é controlado pelo bean HDScanner MC configurado no arquivo **`JBOSS_DIST/jboss-as/server/<instance-name>/deploy/hdscanner-jboss-beans.xml`**. Para a configuração do servidor **`default`** o scanPeriod é ajustado para 5 segundos:

```
<bean name="HDScanner" class="org.jboss.system.server.profileservice.hotdeploy.HDScanner">
 <property name="deployer"><inject bean="ProfileServiceDeployer"/></property>
 <property name="profileService"><inject bean="ProfileService"/></property>
 <property name="scanPeriod">5000</property>
```

```
 <property name="scanThreadName">HDScanner</property>
</bean>
```

O atributo scanPeriod controla o intervalo para a segmentação que reconhece as alterações do hot deployment.

Nota

As alterações do arquivo **`hdscanner-jboss-beans.xml`** por conta própria são hot deployable. Nenhuma reinicialização do servidor é necessária.

1.5.2. Adicionando uma pasta de implementação personalizada

Por padrão, o servidor do JBoss procura por implementações sob a pasta **`JBOSS_DIST/jboss-as/server/<instance-name>/deploy`**. No entanto, você pode configurar o servidor para incluir a sua pasta personalizada em implementações de escaneamento. Isto pode ser realizado pela configuração do bean **`BootstrapProfileFactory`** MC no arquivo **`JBOSS_DIST/jboss-as/server/<instance-name>/conf/bootstrap/profile.xml`**. A propriedade applicationURIs do **`BootstrapProfileFactory`** aceita uma lista de URLs que podem ser escaneados para aplicativos. Você pode adicionar a sua pasta de implementação personalizada para esta lista. Por exemplo, caso você queira que o **`/home/me/myapps`** seja escaneado para implementações, você poderá adicionar o seguinte:

```
<bean name="BootstrapProfileFactory"
 class="org.jboss.system.server.profileservice.repository.
StaticProfileFactory">
 ...
 <property name="applicationURIs">
  <list elementClass="java.net.URI">
   <value>${jboss.server.home.url}deploy</value>
   <value>file:///home/me/myapps</value>
  </list>
 ...
```

Importante

A modificação do **`JBOSS_DIST/jboss-as/server/<instance-name>/conf/bootstrap/profile.xml`** requer a reinicialização do servidor, para que as alterações tenham efeito.

Por motivos de desempenho, a adição de uma nova pasta de implementação ao **`BootstrapProfileFactory`** também solicita que o mesmo URL seja adicionado à configuração do bean **`VFSCache`** MC no **`JBOSS_DIST/jboss-as/server/<instance-name>/conf/bootstrap/vfs.xml`**. Por exemplo:

```
<bean name="VFSCache">
 ...
 <property name="permanentRoots">
  <map keyClass="java.net.URL" valueClass="org.jboss.virtual.spi.ExceptionHandler">
   ...
   <entry>
    <key>file:///home/me/myapps</key>
    <value><inject bean="VfsNamesExceptionHandler"/></value>
   </entry>
```

```
 </map>
</property>
...
```

Importante

A não adição da pasta de implementação personalizada ao **VFSCache** pode resultar num crescimento de usagem do espaço do disco pelo servidor, sobre um período de tempo.

1.6. Problemas Básicos de Configuração

Agora que já examinamos o servidor JBoss, veremos os principais arquivos de configuração e para que são usados. Todos os caminhos são relativos ao diretório de configuração do servidor (**server/default**, por exemplo).

1.6.1. Configurando seu aplicativo como um aplicativo padrão no servidor

Por padrão, o servidor do JBoss configura **JBOSS_DIST/jboss-as/server/<instance-name>/deploy/ROOT.war** como um aplicativo padrão no servidor. Desta forma, o acesso ao **http://localhost:8080/** resultará na exibição da página índex deste aplicativo. Caso você deseje que seu aplicativo esteja disponível como um aplicativo padrão, você deverá seguir os seguintes passos:

- Nomeie novamente o **ROOT.war** no **JBOSS_DIST/jboss-as/server/<instance-name>/deploy** com um outro nome, por exemplo, **jboss.war**.
- No seu arquivo WAR (aquele em que você almeja ser o aplicativo padrão), adicione o **jboss-web.xml** na pasta, com uma configuração para o context-root:

```
<?xml version="1.0"?>
<!DOCTYPE jboss-web PUBLIC "-//JBoss//DTD Web Application 5.0//EN" "http://www.jboss.org/
j2ee/dtd/jboss-web_5_0.dtd">

<jboss-web>
 <context-root>/</context-root>
 <!-- Other configurations as needed -->
</jboss-web>
```

O ajuste do context-root para **/**, fará do seu aplicativo um aplicativo padrão. Agora, seu aplicativo estará disponível no **http://localhost:8080/**.

Nota

A renomeação do **ROOT.war** para **jboss.war** fará com que aquele aplicativo esteja disponível no **http://localhost:8080/jboss**.

1.6.2. Configuração Bootstrap

A configuração bootstrap de microcontainer está descrita no **conf/bootstrap.xml** e **conf/bootstrap/*.xml**. Espera-se que o número de beans bootstrap seja reduzido no futuro. Não espera-se precisar editar os arquivos de configuração bootstrap para uma instalação típica.

1.6.3. Serviços Centrais de Legacia

Os serviços centrais especificados no arquivo **`conf/jboss-service.xml`** são iniciados primeiro quando o servidor é iniciado. Se você observar este arquivo em um editor, verá o MBeans para diversos serviços incluindo a autenticação, segurança, JNDI, JNDIView, etc. Tente comentar a entrada para o serviço **`JNDIView`**.

Nota

Eventualmente, este arquivo será derrubado uma vez que os serviços são convertidos aos beans microcontainer ou mbeans que são implementados como serviços do diretório de implementação.

Note que como a definição dos mbeans aninharam comentários, tivemos que comentar o mbean em duas seções, deixando o comentário original como era.

```
<!-- Section 1 commented out
<mbean code="org.jboss.naming.JNDIView"
    name="jboss:service=JNDIView"
    xmbean-dd="resource:xmdesc/JNDIView-xmbean.xml">
-->
    <!-- The HANamingService service name -->
<!-- Section two commented out
    <attribute name="HANamingService">jboss:service=HAJNDI</attribute></mbean>
-->
```

Se você reiniciar o JBoss, verá que o serviço **`JNDIView`** não aparece mais na listagem do Console de Gerenciamento (Console JMX). Na prática, você não terá que modificar, ou modificará raramente este arquivo. No entanto, não há nada que o impeça de adicionar entradas de MBean extras, caso seja desejável. A alternativa é usar um arquivo separado no diretório **`deploy`**, que permite que seu serviço seja hot deployable.

1.6.4. Serviço de Autenticação

No JBoss, o **`log4j`** é usado para a autenticação. Caso você não seja familiar com o pacote **`log4j`** e gostaria de usar isto em seus aplicativos, você poderá obter mais informações a respeito disto no site da web Jakarta (*http://jakarta.apache.org/log4j/*).

A autenticação é controlada a partir de um arquivo **`conf/jboss-log4j.xml`** central. Este arquivo define um conjunto de anexos especificando os arquivos de autenticação, dos quais as categorias de mensagens devem ser encaminhadas à ele, o formato de mensagem e o nível de filtro. Por padrão, o JBoss produz o resultado de ambos os consoles de um arquivo de autenticação (**`log/server.log`**).

Existem 5 níveis de autenticação básicos usados: **`TRACE`**, **`DEBUG`**, **`INFO`**, **`WARN`**, **`ERROR`** e **`FATAL`**. O valor limite da autenticação no console é **`INFO`**, o que significa que você verá mensagens informacionais, mensagens de aviso e erros no console, porém não verá mensagens de depuração. Por um outro lado, não existe valor limite configurado para o arquivo **`server.log`**, portanto todas as mensagens de autenticação geradas serão autenticadas ali.

Se algo estiver dando errado e parecer não existir nenhuma informação útil no console, verifique o arquivo **`server.log`** para confirmação da existência de qualquer mensagem de depuração que possa ajudá-lo a rastrear o problema. No entanto, tenha em mente que só porque o valor limite de autenticação permite que mensagens de depuração sejam exibidas, isto não significa que todos

os JBoss produzirão informações de depuração detalhadas para o arquivo de autenticação. Você também terá que aumentar os limites de autenticação configurados para categorias individuais. Tome como exemplo a seguinte categoria:

```
<!-- Limit JBoss categories to INFO -->
<category name="org.jboss">
    <priority value="INFO"/>
</category>
```

Isto limita o nível de autenticação do **INFO** para todas as classes do JBoss, fora aquelas que possuem mais substituições fornecidas. Por padrão, o autenticador root no **jboss-log4j.xml** é configurado para **INFO**. Isto significa efetivamente que qualquer autenticador **TRACE** ou **DEBUG** de qualquer categoria de autenticador não será autenticado em quaisquer arquivos ou anexador de console. Esta configuração é controlada através da propriedade jboss.server.log.threshold. Por padrão, isto é **INFO**. Caso você deseje alterar isto para **DEBUG**, isto produzirá um resultado de autenticação muito mais detalhado. Existem duas opções para alteração disto:

- Você pode passar o parâmetro -Djboss.server.log.threshold=DEBUG enquanto inicializando o servidor:

  ```
  ./run.sh -Djboss.server.log.threshold=DEBUG
  ```

- Você pode editar o arquivo **JBOSS_DIST/jboss-as/server/<instance-name>/conf/jboss-log4j.xml** diretamente com o objetivo de configurá-lo apropriadamente:

  ```
  <root>
   <!-- Let's comment this out to set our own value
   <priority value="${jboss.server.log.threshold}"/>-->
   <priority value="DEBUG"/>
   <appender-ref ref="CONSOLE"/>
   <appender-ref ref="FILE"/>
  </root>
  ```

Nota

O **JBOSS_DIST/jboss-as/server/<instance-name>/conf/jboss-log4j.xml** é escaneado a cada 60 segundos (por padrão) para checar por quaisquer alterações. A alteração deste arquivo não requer a reinicialização do servidor uma vez que as alterações não serão hot deployed nos próximos 60 segundos seguintes da mudança.

Um outro exemplo, digamos que você queira ajustar o resultado a partir da máquina de persistência de container gerenciado para o nível **DEBUG** e redirecioná-lo para um arquivo separado, **cmp.log**, para que possa analisar os comandos SQL gerados. Você teria que adicionar o seguinte código ao arquivo **conf/jboss-log4j.xml**:

```
<appender name="CMP" class="org.jboss.logging.appender.RollingFileAppender">
    <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
    <param name="File" value="${jboss.server.home.dir}/log/cmp.log"/>
    <param name="Append" value="false"/>
    <param name="MaxFileSize" value="500KB"/>
    <param name="MaxBackupIndex" value="1"/>
```

```
    <layout class="org.apache.log4j.PatternLayout">
        <param name="ConversionPattern" value="%d %-5p [%c] %m%n"/>
    </layout>
</appender>

<category name="org.jboss.ejb.plugins.cmp">
    <priority value="DEBUG" />
    <appender-ref ref="CMP"/>
</category>
```

Isto cria um novo anexador de arquivo e especifica que deve ser usado pelo autenticador (ou categoria) para o pacote **`org.jboss.ejb.plugins.cmp`**.

O anexador de arquivos está configurado para produzir um novo arquivo de autenticação todos os dias, ao invés de produzir um novo todas as vezes que você reiniciar o servidor ou gravar em um arquivo por um tempo indeterminado. O arquivo de registro atual é **`cmp.log`**. Arquivos mais antigos, possuem a data que foram gravados adicionado ao nome. Você notará que o diretório **`log`** também contém o registro de requisição HTTP que são produzidos pelo container da Web.

Por padrão, o anexador **`server.log`** é configurado para manter as mensagens de autenticação entre a reinicialização do servidor. Isto é controlado pela propriedade Append no anexador **`FILE`** que corresponde ao arquivo **`server.log`**. Esta propriedade é configurada para verdadeiro por padrão. Caso você deseje que os conteúdos **`server.log`** sejam apagados na reinicialização do servidor, você poderá editar o arquivo **`JBOSS_DIST/jboss-as/server/<instance-name>conf/jboss-log4j.xml`** para ajustar este valor de propriedade para falso. Por exemplo:

```
<appender name="FILE" class="org.jboss.logging.appender.DailyRollingFileAppender">
 <errorHandler class="org.jboss.logging.util.OnlyOnceErrorHandler"/>
  <param name="File" value="${jboss.server.log.dir}/server.log"/>
  <param name="Append" value="false"/>
  ...
```

1.6.5. Serviço de Segurança

A informação do domínio de segurança é armazenada no arquivo **`conf/login-config.xml`** como uma lista de domínios de segurança nomeados, cada qual especificando um número do JAAS [2] módulos de logon que são usados para propósitos de autenticação naquele domínio. Quando quiser usar a segurança em um aplicativo, especifique um nome de domínio que desejar para usar nos descritores de implementação do JBoss específico do aplicativo, **`jboss.xml`** (usado na definição das configurações específicas do jboss) e/ou **`jboss-web.xml`** (usadas na definição do jboss para o aplicativo da Web. Veremos rapidamente como fazer isto para proteger o aplicativo do Console JMX que distribui o JBoss.

Quase todos os aspectos do servidor do JBoss podem ser controlados através do Console JMX, portanto é importante que você tenha certeza de que, pelo menos, a senha do aplicativo esteja protegida. Do contrário, qualquer usuário remoto poderia ter o total controle de seu servidor. Para protegê-lo, nós adicionaremos um domínio de segurança para cobrir o aplicativo. Isto pode ser realizado no arquivo **`jboss-web.xml`** do Console JMX, que pode ser encontrado no diretório **`deploy/jmx-console.war/WEB-INF/`**. Descomente o **`security-domain`** naquele arquivo, como mostrado abaixo.

```
<jboss-web>
```

[2] O Serviço de Autorização e Autenticação do Java. O JBoss usa o JAAS para oferecer módulos de autenticação plugáveis. Você pode usar aqueles que são fornecidos ou gravar seu próprio, caso tenha requerimentos mais específicos.

```
    <security-domain>java:/jaas/jmx-console</security-domain>
</jboss-web>
```

Isto liga o domínio de segurança ao aplicativo da web, mas não informa o aplicativo da Web sobre qual política de segurança impor, quais URLs estamos tentando proteger, e quem possui acesso à eles. Para configurar isto, vá ao arquivo **`web.xml`** no mesmo diretório e descomente o **`security-constraint`** que já está lá. Esta restrição de segurança irá requerer um nome de usuário válido e uma senha para um usuário no grupo **`JBossAdmin`**.

```
<!--
   A security constraint that restricts access to the HTML JMX console
   to users with the role JBossAdmin. Edit the roles to what you want and
   uncomment the WEB-INF/jboss-web.xml/security-domain element to enable
   secured access to the HTML JMX console.
-->
<security-constraint>
    <web-resource-collection>
        <web-resource-name>HtmlAdaptor</web-resource-name>
        <description>
            An example security config that only allows users with the
            role JBossAdmin to access the HTML JMX console web application
        </description>
        <url-pattern>/*</url-pattern>
        <http-method>GET</http-method>
        <http-method>POST</http-method>
    </web-resource-collection>
    <auth-constraint>
        <role-name>JBossAdmin</role-name>
    </auth-constraint>
</security-constraint>
```

Ótimo, mas de onde os nomes de usuário e senhas vieram? Eles vieram do domínio de segurança **`jmx-console`** ao qual conectamos o aplicativo. Nós fornecemos a configuração para ele no **`conf/login-config.xml`**.

```
<application-policy name="jmx-console">
    <authentication>
        <login-module code="org.jboss.security.auth.spi.UsersRolesLoginModule"
                      flag="required">
            <module-option name="usersProperties">
                props/jmx-console-users.properties
            </module-option>
            <module-option name="rolesProperties">
                props/jmx-console-roles.properties
            </module-option>
        </login-module>
    </authentication>
</application-policy>
```

Esta configuração usa uma política de segurança baseada em arquivo simples. Os arquivos de configuração são encontrados no diretório **`conf/props`** de sua configuração de servidor. Os nomes de usuários e senhas são armazenados no arquivo **`conf/props/jmx-console-users.properties`** e levam o "**`username=password`**". Para atribuir um usuário ao grupo **`JBossAdmin`** adicione o "**`username=JBossAdmin`**" ao arquivo **`jmx-console-roles.properties`** (as funções adicionais do nome do usuário podem ser adicionadas separadas por vírgula). O arquivo existente cria um usuário **`admin`** com uma senha **`admin`**. Por segurança, por favor remova o usuário ou mude a senha para uma senha mais confiável.

O JBoss irá reimplementar o Console JMX quando você atualizar seu **`web.xml`**. Você pode verificar o console do servidor para confirmar que o JBoss já viu estas mudanças. Se você configurou tudo

corretamente e reimplementou o aplicativo, da próxima vez que você tentar acessar o Console JMX, ele lhe solicitará o nome e a senha. [3]

O Console JMX não é a única interface de gerenciamento baseada na Web para o JBoss. Existe também o Console da Web. Apesar de ser um applet do Java, o aplicativo da Web correspondente pode ser protegido da mesma forma que o Console JMX. O Console da Web está no **`deploy/management/web-console.war`** do arquivo. A única diferença é que o Console da Web é fornecido como um simples arquivo WAR ao invés de usar a estrutura do diretório destacado que o Console JMX fez. A única real diferença entre os dois é que editar os arquivos dentro do arquivo WAR é um pouco mais complicado.

1.6.6. Serviços Adicionais

Serviços não centrais e hot-deployable são adicionados ao diretório **`deploy`**. Eles podem ser tanto arquivos descritores XML **`*-service.xml, *-jboss-beans.xml`**, MC **`.beans`** ou arquivos do JBoss Service Archive (SAR). Os SARs contém um descritor **`META-INF/jboss-service.xml`** e os recursos adicionais que o serviço requer (ex.: classes, arquivos de bibliotecas JAR, ou outros arquivos), todos empacotados em um só arquivo. Adicionado a isto, o arquivo **`.beans`** contém um **`META-INF/jboss-beans.xml`** e recursos adicionais.

Informações detalhadas sobre todos estes serviços podem ser encontradas na *Plataforma do Aplicativo JBoss Enterprise: Guia de Administração e Configuração*, que também oferece informação compreensiva sobre servidores internos e implementação de serviços tais como JTA e Arquitetura do Conector J2EE - J2EE Connector Architecture (JCA).

1.7. Gerenciador de Vinculação do Serviço

O servidor JBoss usa vários portais para serviços que isto fornece (por exemplo, portal 8080 para HTTP, 1099 para JNDI). O serviço do Gerenciador de Vinculação do Serviço - Service Binding Manager (SBM) fornece uma localização centralizada onde as configurações de todos os serviços que precisam vincular os portais possam ser configuradas. O SBM pode ser usado para configurar conjuntos diferentes de vinculações de portais para uma instância do servidor. Uma propriedade do sistema no SBM controla qual conjunto nomeado (por exemplo, ports-default, ports-01) é usado por uma instância do servidor particular. Caso você queira rodar instâncias do servidor múltiplas no mesmo sistema, você poderá configurar o SBM em cada instância para uso de um conjunto diferente de vinculação nomeado. Você pode ainda usar o SBM para alterar um conjunto de vinculação diferente (por exemplo, portal 8180 para HTTP ao invés do padrão 8080) para uma instância do servidor.

Numa configuração típica, o conjunto **`ports-default`** usa os portais padrões (por exemplo, JNDI no 1199), com o **`ports-01`** aumentando cada valor de portal para 100 (por exemplo, JNDI no 1199), o **`ports-02`** para 200 e assim por diante.

O SBM é configurado através do arquivo **`$JBOSS_DIST/jboss-as/server/<instance-name>/conf/bindingservice.beans/ META-INF/bindings-jboss-beans.xml`**. A configuração do **`ServiceBindingManager`** envolve os três elementos primários:

- Um conjunto de beans contendo os dados de configuração de vinculação padrão (default). Estes são valores básicos (por exemplo, JNDI no 1099) usados para orientar o **`ports-default`**, **`ports-01`** e assim por diante.

[3] Uma vez que o nome do usuário e senha são sessões variáveis no do navegador da web, talvez seja necessário reiniciar o seu navegador para uso da janela de diálogo de logon.

- Um número de beans definindo **`ServiceBindingSets`**, por exemplo, **`ports-default`**, **`ports-01`**, **`ports-02`**. Os conjuntos das vinculações padrões são combinadas com cada um deles, através de um valor (por exemplo, 100 para **`ports-01`**) que deve ser aplicado aos valores do portal padrão para criar valores de vinculação para o conjunto.
- O bean de serviço **`ServiceBindingManager`** por conta própria. Ele possui as vinculações padrões e o **`ServiceBindingSets`** injetado a isto. Ele também é configurado com o nome do conjunto de vinculação que uma instância de servidor particular deve usar. O nome do conjunto de vinculação a ser usado é configurável a partir da linha de comando usando o jboss.service.binding.set da propriedade de sistema. O valor padrão é **`ports-default`**.

```
<bean name="ServiceBindingManagementObject"
 class="org.jboss.services.binding.managed.ServiceBindingManagementObject">
 <constructor>

  <parameter>
   ${jboss.service.binding.set:ports-default}
  </parameter>
 ...
```

Para alterar a um conjunto diferente de portais que aqueles usados por padrão, você pode iniciar o servidor passando a propriedade para o comando rodar, conforme abaixo:

```
./run.sh -Djboss.service.binding.set=ports-01
```

Isto orientará o servidor a usar o grupo de portais configurados no conjunto de vinculação **`ports-01`**.

Usando outros Bancos de Dados

Nos capítulos anteriores, nós estávamos usando o banco de dados padrão do servidor da Plataforma do Aplicativo JBoss Enterprise em nossos aplicativos. Este banco de dados é configurado para uso da instância do banco de dados Hypersonic incorporado com a distribuição. Este banco de dados é vinculado ao nome JNDI **`java:/DefaultDS`** e seu descritor é nomeado **`hsqldb-ds.xml`** sob o diretório de implementação.

Possuir um banco de dados incluído com a Plataforma do Aplicativo JBoss Enterprise é bastante conveniente para rodar servidores e amostras out-of-the-box. No entanto, este banco de dados não é um banco de dados de qualidade de produção, de tal forma que ele não deve ser usado com as implementações de classe empresarial. Como consequência disto, o Suporte do JBoss não fornece qualquer suporte oficial para o Hypersonic.

Neste capítulo, nós explicaremos em detalhes como configurar e implementar um banco de dados para conectar a Plataforma do Aplicativo JBoss Enterprise aos servidores do banco de dados mais populares atualmente no mercado.

2.1. Arquivos de Configuração de Fonte de Dados

Os nomes de arquivos de configuração da Fonte de Dados terminam no sufixo **`-ds.xml`**, portanto serão reconhecidos corretamente pelo implementador JCA. O diretório **`docs/example/jca`** contém arquivos de amostra para uma seleção ampla de banco de dados e é uma ótima ideia usá-los como ponto inicial. Para uma descrição completa do formato da configuração, consulte o arquivo DTD **`docs/dtd/jboss-ds_1_5.dtd`**. Documentos adicionais neste arquivo e na implementação JCA do JBoss podem também ser encontrados na Administração da Plataforma do Aplicativo JBoss Enterprise e Guia de Configuração do Servidor disponíveis no *http://www.redhat.com/docs/en-US/JBoss_Enterprise_Application_Platform/*.

As fontes de dados da transação local são configuradas usando o elemento **`local-tx-datasource`** e aqueles compatíveis com o XA usando **`xa-tx-datasource`**. O arquivo exemplo **`generic-ds.xml`** demonstra como usar ambos os tipos e também alguns outros elementos que estão disponíveis para configuração de conexão em pool, entre outras ações. Exemplos de configurações de XA e local encontram-se disponíveis no Oracle, DB2 e Informix.

Se você observar os arquivos de amostra **`firebird-ds.xml`**, **`facets-ds.xml`** e **`sap3-ds.xml`**, irá notar que eles possuem um formato totalmente diferente, com um elemento root sendo **`connection-factories`** ao invés de **`datasources`**. Estes usam uma sintaxe de configuração JCA mais genérica e alternativa, usada com um adaptador de recurso JCA pré-empacotado. A sintaxe não é específica para a configuração da fonte de dados e é usada, por exemplo, no arquivo **`JBOSS_DIST/jboss-as/server/<instance-name>/deploy/messaging/jms-ds.xml`** para configurar o adaptador de recurso JMS.

Em seguida, passaremos aos exemplos detalhadamente para ilustrar a montagem de uma fonte de dados para um banco de dados específico.

2.2. Usando o MySQL como Fonte de Dados Padrão

O banco de dados MySQL® é um dos bancos de dados de código aberto mais populares graças a sua consistência de desempenho rápido e facilidade de uso. Este servidor de banco de dados é usado em milhares de instalações variando de grandes corporações a aplicativos de incorporação especializados em todo o mundo. O driver JDBC oficial é chamado **`Connector/J`**. Nós usamos o MySQL 5.1.31 e **`Connector/J`** 5.1.8 para esta amostra. Você pode baixá-los a partir do *http://www.mysql.com*.

2.2.1. Criando um Banco de Dados e um Usuário

Partimos do pressuposto de que você já possui o MySQL instalado e em execução, além de você já está familiarizado com suas funções básicas. Execute o programa cliente mysql a partir da linha de comando, assim poderemos executar alguns comandos de administração. Você precisa ter certeza de que está conectado como um usuário que possui privilégios suficientes (ex. especificando a opção root **-u** para rodar como usuário root do MySQL).

Primeiro crie um banco de dados chamado jboss dentro do MySQL para uso do JBoss:

```
mysql> CREATE DATABASE jboss;

Query OK, 1 row affected (0.05 sec)
```

Depois, confirme se ele foi mesmo criado:

```
mysql> SHOW DATABASES;

+----------+
| Database |
+----------+
| jboss    |
+----------+
1 rows in set (0.00 sec)
```

Em seguida, crie um usuário chamado jboss com a senha password para acessar o banco de dados:

```
mysql> GRANT ALL PRIVILEGES ON jboss.* TO jboss@localhost IDENTIFIED BY 'password';

Query OK, 0 rows affected (0.06 sec)
```

Novamente, verifique se tudo está de acordo:

```
mysql> select User,Host,Password from mysql.User;

+-------+-----------+------------------+
| User  | Host      | Password         |
+-------+-----------+------------------+
| root  | localhost |                  |
| root  | %         |                  |
|       | localhost |                  |
|       | %         |                  |
| jboss | localhost | 5d2e19393cc5ef67 |
+-------+-----------+------------------+
5 rows in set (0.02 sec)
```

2.2.2. Instalando o Driver JDBC e Desenvolvendo a Fonte de Dados

Para disponibilizar as classes do driver JDBC para a Plataforma do Aplicativo JBoss Enterprise, copie o arquivo **mysql-connector-java-5.1.8-bin.jar** da distribuição **Connector/J** para o diretório **lib** na configuração do servidor **default** (considerando que você esteja rodando esta configuração, é claro).

Crie um arquivo no diretório de implementação chamado **mysql-ds.xml** com a seguinte configuração do banco de dados. Perceba que o nome do usuário e senha do banco de dados corresponde ao usuário MySQL que criamos na seção anterior:

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
 <local-tx-datasource>
  <jndi-name>MySqlDS</jndi-name>
  <connection-url>jdbc:mysql://localhost:3306/jboss</connection-url>
  <driver-class>com.mysql.jdbc.Driver</driver-class>
  <user-name>jboss</user-name>
  <password>password</password>
 </local-tx-datasource>
</datasources>
```

Para garantir que você configurou o banco de dados corretamente na pasta **`JBOSS_DIST/jboss-as/server/<instance-name>/deploy`**, inicie o servidor e você perceberá mensagens como estas na autenticação:

```
INFO  [ConnectionFactoryBindingService] Bound ConnectionManager
 'jboss.jca:service=DataSourceBinding,name=MySqlDS' to JNDI name 'java:MySqlDS'
```

Nota

A configuração de outras fontes de dados é um processo parecido.

2.2.3. Testando a Fonte de Dados do MySQL

Você verificará uma instalação apropriada de sua fonte de dados com o uso do cliente teste descrito na *Seção 2.6, "Criando um cliente JDBC"*.

2.3. Configurando uma fonte de dados para o Oracle DB

O Oracle é um dos participantes principais no campo de banco de dados comercial e a maioria dos leitores terão acesso a ele em algum tempo determinado. Você pode baixá-lo livremente com propósitos não comerciais a partir do site *http://www.oracle.com/technology/products/database/xe/index.html*.

Nesta seção, conectaremos o servidor na Edição Expressa de 11g do Banco de Dados Oracle usando o último lançamento do driver JDBC (11g) disponível no *http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html*.

2.3.1. Instalando o Driver JDBC e Desenvolvendo a Fonte de Dados

Para fazer com que as classes do driver JDBC estejam disponíveis à Plataforma do Aplicativo JBoss Enterprise, copie o arquivo **`ojdbc6.jar`** ao diretório lib na configuração do servidor padrão (assumindo que esta é a configuração do servidor que você está rodando).

Depois disso, crie um arquivo de texto no diretório **`deploy`** chamado **`oracle-ds.xml`**, com o seguinte descritor da fonte de dados:

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
 <local-tx-datasource>
  <jndi-name>DefaultDS</jndi-name>
  <connection-url>jdbc:oracle:thin:@localhost:1521:xe</connection-url>
  <driver-class>oracle.jdbc.driver.OracleDriver</driver-class>
  <user-name>SYSTEM</user-name>
```

```
  <password>jboss</password>
  <valid-connection-checker-class-
name>org.jboss.resource.adapter.jdbc.vendor.OracleValidConnectionChecker</valid-connection-
checker-class-name>
  <metadata>
   <type-mapping>Oracle9i</type-mapping>
  </metadata>
 </local-tx-datasource>
</datasources>
```

A fonte de dados está apontando ao banco de dados/SID chamado *xe*, fornecido por padrão com Oracle XE.

Você precisa atualizar os atributos url de conexão assim como a combinação de nome de usuário/ senha para combinar sua configuração do ambiente.

2.3.2. Testando a Fonte de Dados do Oracle

Antes de você poder verificar a configuração da fonte de dados, o Oracle XE deve ser reconfigurado para evitar conflito de portais da Plataforma do Aplicativo JBoss Enterprise uma vez que, por padrão, ambos iniciam num servidor da web no portal 8080.

Abra uma linha de comando do Oracle SQL e execute os seguintes comandos:

```
SQL> connect; Enter user-name: SYSTEM Enter password:
Connected.
SQL> begin 2 dbms_xdb.sethttpport('8090'); 3 end; 4 /
PL/SQL procedure successfully completed.
SQL> select dbms_xdb.gethttpport from dual;
GETHTTPPORT
-----------
8090
```

O servidor da web iniciado pelo Oracle XE para fornecer as ferramentas de administração baseada no http, está rodando agira no portal 8090. Inicie normalmente a instância do servidor da Plataforma do Aplicativo JBoss Enterprise. Agora você está pronto para usar o cliente teste com o objetivo de verificar a instalação própria de sua fonte de dados.

2.4. Configurando a fonte de dados para o Microsoft SQL Server 200x

Nesta seção, nós conectaremos o servidor para o MS SQL Server 2005 usando o JDBC driver (v2.0) mais avançado disponível no *http://msdn2.microsoft.com/en-us/data/aa937724.aspx*.

2.4.1. Instalando o Driver JDBC e Desenvolvendo a Fonte de Dados

Para disponibilizar as classes do driver JDBC na Plataforma do Aplicativo JBoss Enterprise, copie o **`sqljdbc.jar`** do arquivo a partir da distribuição **`sqljdbc_2.0`** para o diretório **`lib`** na configuração do servidor padrão (assumindo que isto é uma configuração que você está executando).

Em seguida, crie um arquivo de texto no diretório **`deploy`** chamado **`mssql-ds.xml`**, com o seguinte descritor da fonte de dados:

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
  <local-tx-datasource>
  <jndi-name>DefaultDS</jndi-name>
```

```
  <connection-url>jdbc:sqlserver://localhost:1433;DatabaseName=pubs</connection-url>
  <driver-class>com.microsoft.sqlserver.jdbc.SQLServerDriver</driver-class>
  <user-name>sa</user-name>
  <password>jboss</password>
  <check-valid-connection-sql>SELECT 1 FROM sysobjects</check-valid-connection-sql>
  <metadata>
   <type-mapping>MS SQLSERVER2000</type-mapping>
  </metadata>
  </local-tx-datasource>
</datasources>
```

A fonte de dados está apontando ao banco de dados **pubs**, fornecido por padrão com o MS SQL Server 2000.

Lembre-se de atualizar os atributos url de conexão assim como a combinação de nome de usuário/ senha para combinar sua configuração de ambiente.

2.4.1.1. Testando a Fonte de Dados

Você verificará uma instalação apropriada de sua fonte de dados com o uso do cliente teste descrito na *Seção 2.6, "Criando um cliente JDBC"*.

2.5. Configurando o Gerenciador de Persistência do JBoss Messaging

O gerenciador de persistência do JBoss Messaging usa a fonte de dados para criar tabelas para o armazenamento de mensagens, dados de transação e outros indexes. A configuração de "persistência" está agrupada nos arquivos **`xxx-persistence-service.xml`**. A Plataforma do Aplicativo JBoss Enterprise distribui um arquivo **`hsqldb-persistence-service.xml`** padrão, do qual configura o servidor Messaging para usar a instância do banco de dados Hypersonic que distribui a Plataforma do Aplicativo JBoss Enterprise, por padrão.

Você pode visualizar o arquivo **`hsqldb-persistence-service.xml`** nas configurações baseadas nas configurações *all* ou *default*:

```
<JBoss_Home>/server/all/deploy/messaging/hsqldb-persistence-service.xml  and
   <JBoss_Home>/server/default/deploy/messaging/hsqldb-persistence-service.xml
```

Aviso

O banco de dados Hypersonic não é recomendado para ambientes de produção devido ao seu suporte limitado para a transação isolada e sua própria segurança sob carga pesada.

Maiores informações sobre a configuração do JBoss Messaging podem ser encontradas no *Guia de Administração e Configuração*[1].

2.6. Criando um cliente JDBC

Sugerimos o uso de alguns códigos de clientes JDBC básicos incorporados na página JSP, quando testando um banco de dados recém-configurado. Antes de mais nada, você deverá criar um

[1] http://www.redhat.com/docs/en-US/JBoss_Enterprise_Application_Platform/5.0.0/html-single/Administration_And_Configuration_Guide/index.html

arquivo WAR destacado sob o diretório de implementação que é, na realidade, uma pasta nomeada "**`jdbcclient.war`**". Nesta pasta, crie um documento de texto nomeado client.jsp e cole o código seguinte:

```
<%@page contentType="text/html"
 import="java.util.*,javax.naming.*,javax.sql.DataSource,java.sql.*"
 %>
 <%

  DataSource ds = null;
  Connection con = null;
  PreparedStatement pr = null;
  InitialContext ic;
  try {
  ic = new InitialContext();
  ds = (DataSource)ic.lookup( "java:/DefaultDS" );
  con = ds.getConnection();
  pr = con.prepareStatement("SELECT USER_ID, PASSWD FROM JBM_USER");
  ResultSet rs = pr.executeQuery();
  while (rs.next()) {
  out.println("<br> " +rs.getString("USER_ID") + " | " +rs.getString("PASSWD"));
  }
  rs.close();
  pr.close();
  }catch(Exception e){
  out.println("Exception thrown " +e);
  }finally{
  if(con != null){
  con.close();
 }
} %>
```

Abra um navegador da web e clique url: *http://localhost:8080/jdbcclient/client.jsp*. Uma lista de usuários e senhas deverá aparecer como resultado da consulta JDBC:

```
dynsub | dynsub
guest | guest
j2ee | j2ee
john | needle
nobody | nobody
```

Apêndice A. Histórico de Revisão

Revisão 1.2 **Thu Oct 29 2009** **Laura Bailey** *lbailey@redhat.com*

correções do JIRA.