JBoss Enterprise Application Platform 5

スタートガイド

JBoss Enterprise Application Platform 5 ユーザー向け

エディッション 5.1.2

Last Updated: 2017-10-26

JBoss Enterprise Application Platform 5 スタートガイド

JBoss Enterprise Application Platform 5 ユーザー向け
エディッション 5.1.2

Red Hat ドキュメンテーショングループ

法律上の通知

概要

本スタートガイドには、JBoss Enterprise Application Platform 5 を初めて使用する場合に有用な情報が記載されています。

目次

第1章 JBOSS サーバー - クイックツアー

1.1. サーバー構成

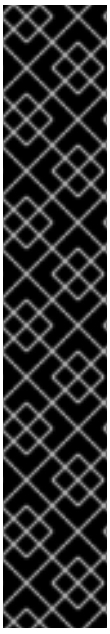
アプリケーションサーバーの構造の詳細については、 このリリースの JBoss Enterprise Application Platform に付属する『インストールガイド (Installation Guide)』の章「移行 (Migration)」を参照してください。.

1.2. サーバーの開始と停止

1.2.1. サーバーの開始

***JBOSS_DIST*/jboss-as/bin** ディレクトリに移動し、**run.sh** スクリプト (Linux用) あるいは**run.bat** スクリプト (Microsoft Windows 用) を実行します。

production プロファイルを使用してサーバーが開始された場合、コンソールには**Server Started** のメッセージは表示されません。 このメッセージは、 **server.log** file located in the ***JBOSS_DIST*/jboss-as/server/production/logs/log** サブティレクトリにて確認できます。



重要

JBoss JBoss Enterprise Application Platform は、使用可能な全てのインターフェース (0.0.0.0) にバインドするのではなく、デフォルトでサービスをローカルホスト (127.0.0.1) にバインドするようになりました。 これは、 主にユーザーがサーバーを適切に保護せずに実稼働する懸念があるというセキュリティ上の理由のためです。 JBoss サービスを特定のインターフェースにバインドし、 リモートアクセスを有効にするには、 **-b** オプションを用いて JBoss を実行します。 使用可能な全てのインターフェースにバインドし、レガシー動作を再度有効にするには、**./run.sh -b 0.0.0.0** (Linux の場合) を使用します。 どちらの場合でも、サーバーを適切に保護する必要があります。

-b を JBoss Server のコマンドラインで使用することは、-Djboss.bind.address、-Djava.rmi.server.hostname、-Djgroups.bind_addr 、-Dbind.address のプロパティを設定すること同じです。 -Djboss.bind.address は JVM プロパティでなく JBoss プロパティであるため、run スクリプトの **JAVA_OPTS** 変数の一部として -Djboss.bind.address を Java プロセスに渡しても、動作しません。

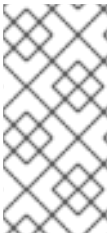
1 台のマシン上に複数の JBoss サーバーインスタンスを設定する方法や、JBoss で複数のドメインをホストする方法などの詳細は、管理設定ガイド を参照してください。

サーバーが起動すると、以下のような画面が出力され (インストールディレクトリの違いは考慮してください)、エラーや例外メッセージは出力されないはずです。

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

  JBoss Bootstrap Environment

  JBOSS_HOME: JBOSS_DIST/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: JBOSS_DIST/jboss-as/bin/run.jar

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

JBoss Enterprise Application Platform の **run** スクリプトに使用できるその他のオプションについては、「代替設定でのサーバーの開始」以下で説明されています。



注記

production プロファイルを使用してサーバーが開始された場合、 コンソールには **Server Started** のメッセージは表示されません。 このメッセージは、 **server/production/log** サブディレクトリにある **server.log** ファイル内で確認できます。

1.2.2. 代替設定でのサーバーの開始

引数なしで **run.sh** を使用すると、 **default** サーバープロファイルファイルセットを使用してサーバーの開始をします。 別のプロファイルファイルセットで開始するには、 **-c** コマンドラインオプションへの値として、 使用したいサーバーの設定ファイルセットの名前 (***JBOSS_DIST*/jboss-as/server/*PROFILE*/** 下のサーバー設定ディレクトリ名と同じ) を渡します。例えば、 **minimal** プロファイルファイルセットで開始するには、以下のように指定します。

```
[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. run.sh あるいは run.bat の使用

run スクリプトは以下のオプションに対応します:

```
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
-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. サーバーの停止

サーバーを終了するには、JBoss が起動されたしたコンソール内で **Ctrl-C** の組み合わせを押します。代わりに **shutdown.sh** コマンドを使用することもできます。

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

shutdown スクリプトは以下のオプションに対応します:

```
A JMX client to shutdown (exit or halt) a remote JBoss server.

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
```

shutdown コマンドの使用には、 **jmx-invoker-service.xml** サービスを含むサーバー設定が必要になります。 そのため、 **minimal** プロファイルでは、shutdown コマンドは使用できません。

1.2.5. システムサービスとして実行

Windows、Linux、および UNIX でアプリケーションサーバーをサービスとして実行できます。手順については、『JBoss Enterprise Application Platform Installation Guide 』 の 『インストール後』 に関する章を参照してください。

1.3. JMX コンソール

JBoss サーバーが稼働している場合は、http://localhost:8080/jmx-console の JMX コンソールアプリケーションにアクセスしてサーバーのライブビューを取得できます。

デフォルトでは、JMX コンソールはセキュア化され、ユーザー名とパスワードが尋ねられます。グラフィカルインストーラーを使用して JBoss Enterprise Application Platform をインストールし、JMX コンソールにアクセスする場合は、インストール時に提供したユーザー名とパスワードを使用できます。

.zip などの他のモードを使用してインストールした場合は、***JBOSS_DIST*/jboss-as/server/*PROFILE*/conf/props/directory** に移動し、**jmx-console-users.properties** ファイル内の管理者ユーザー ID とパスワードのコードをコメント解除します。必要に応じて他のユーザーを追加できます。これにより、定義されたユーザーが **jmx-console-users.properties** ファイル内で指定されたユーザー名とパスワードの組み合わせを使用して JMX コンソールにアクセスできるようになります。

JBoss Enterprise Application Platform のセキュリティサービスの詳細については、「セキュリティサービス」 を参照してください。

重要

サーバー稼働時に **jmx-console-users.properties** ファイルを変更した場合、サーバーを再起動して変更を反映する必要がある場合があります。場合によっては、レイジーローディングにより、サーバーを再起動せずにライブでこの変更を行うことができます。

JMX コンソールは、 サーバーを構成する JMX MBeans の生の表示を提供する JBoss Management Console です。 実行中のサーバーに関する多くの情報を提供する他、 その設定の変更、 コンポーネントの開始と停止などができるようになります。

例えば、 **service=JNDIView** リンクを探してクリックしてみてください。 この特定の MBean はサーバー内の JNDI 名前空間の構造を表示できるようにするサービスを提供します。 今度は、 MBean 表示ページの下部にある **list** と言う演算を見つけて、 **invoke** ボタンをクリックしてみてください。 この演算は JNDI ツリーにバインドされている現在の名前の表示を返します。 独自のアプリケーションの導入を開始した時に特定の EJB 名が解決できない理由を知りたい場合に便利です。

その他の MBeans および一覧表示された演算を見てみます。 一部の設定属性を変更してどうなるか見てみます。 ほんの少しの例外はありますが、 コンソールを介した変更はいずれも永続にはなりません。 JBoss が再起動すると元の設定が再ロードされるので、 永久的なダメージを与えることなく自由に実験することができます。

1.4. JNDIVIEW サービス

JBoss Enterprise Application Platform では、JNDIView サービスがデフォルトで有効となっています。 このサービスは **jmx-console** (http://localhost:8080/jmx-console) の一覧にあります。 **jboss:service=JNDIView** Mbean に移動し、そのリンクをクリックします。MBean 操作ページに **list** メソッドがあります。 **list** メソッドの横にある **Invoke** ボタンをクリックします。

この list 操作により、JNDI ツリーの内容が表示されます。 出力例は下記の通りです。

```
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.TransactionSynchronizat
ionRegistryImple)
+- 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)
```

EJB がバインドされた JNDI 名の詳細が記載されています。

1.5. JBOSS でのサービスのホットデプロイメント

ホットデプロイメントが可能なサービスとは、 稼働中のサーバー上で追加や削除ができるサービスのことです。 これらサービスは、***JBOSS_DIST*/jboss-as/server/<instance-name>/deploy** ディレクトリに配置されています。 JBoss におけるサービスのホットデプロイメントの実例を見てみましょう。

まだ JBoss を起動していない場合は JBoss を起動し、 **server/default/deploy** ディレクトリを見てみます。 **mail-service.xml** ファイルを削除してサーバーからの出力を確認します。

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

次に、ファイルを置き換え、JBoss がサービスを再インストールすることを確認します。

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

これはホットデプロイメントが有効なことを示します。

1.5.1. ホットデプロイメントの設定

サーバー内にあるサービスのホットデプロイメントは、 ***JBOSS_DIST*/jboss-as/server/conf/deploy/hdscanner-jboss-beans.xml** ファイルに設定された HDScanner MC Bean によって制御されます。 **default** サーバー設定の場合、scanPeriod は 5 秒に設定されます。

```
<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>
```

scanPeriod 属性は、ホットデプロイ可能な変更を探知するスレッドの周期を制御します。

注記

hdscanner-jboss-beans.xml ファイルへの変更自体はホットデプロイ可能ではありません。 サーバーの再起動は必要ありません。

1.5.2. カスタムデプロイフォルダの追加

デフォルトでは、JBoss サーバーは ***JBOSS_DIST*/jboss-as/server/<instance-name>/deploy** フォルダー以下のデプロイメントを検索しますが、サーバーを設定してデプロイメントのスキャンにカスタムフォルダーを追加することもできます。 これには、***JBOSS_DIST*/jboss-as/server/*PROFILE*/conf/bootstrap/profile.xml** ファイルの **BootstrapProfileFactory** MC Bean を設定します。 **BootstrapProfileFactory** の applicationURIs プロパティは、スキャンされるアプリケーションの URL 一覧を許可します。 カスタムデプロイフォルダーをこのリストに追加することができます。例えば、**/home/me/myapps** をデプロイメントのスキャンの対象とするには、以下を追加します。

```
<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>
   ...
```

重要

***JBOSS_DIST*/jboss-as/server/*PROFILE*/conf/bootstrap/profile.xml**の変更を有効にするには、サーバーを再起動する必要があります。

パフォーマンス上の理由により、**BootstrapProfileFactory** に新しいデプロイメントフォルダーを追加する際には、***JBOSS_DIST*/jboss-as/server/*PROFILE*/conf/bootstrap/vfs.xml** の **VFSCache** MC Bean 設定に同じ URL を追加する必要があります。 例は次の通りです。

```
<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>
  ...
```

重要

VFSCache にカスタムデプロイメントフォルダを追加しないと、サーバーによるディスクスペースの使用量が長期的に増加することがあります。

1.6. 設定に関する基本的な問題

ここまでは JBoss サーバーの動作を見てきました。 次に主要な設定ファイルをいくつか見てから、 その使用目的を理解します。 すべてのパスはサーバー設定ディレクトリ (例: **server/production**) への相対パスです。

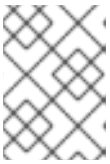
1.6.1. アプリケーションをサーバーのデフォルトアプリケーションとして設定

デフォルトでは、JBoss サーバーは ***JBOSS_DIST*/jboss-as/server/*PROFILE*/deploy/ROOT.war** をサーバー上のデフォルトアプリケーションとして設定します。 そのため、 **http://localhost:8080/** にアクセスすると、アプリケーションのインデックスページが表示されます。 別のアプリケーションをデフォルトのアプリケーションとするには、次の手順に従ってください。

- **ROOT.war** in ***JBOSS_DIST*/jboss-as/server/*PROFILE*/deploy** の名前を変更します (例: **jboss.war**)。
- デフォルトとして使用したいアプリケーションの WAR ファイル内で、**WEB-INF** に **jboss-web.xml** とコンテキストルートの設定を追加します。

```
<?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>
```

コンテキストルートを **/** に設定すると、デフォルトアプリケーションにすることができます。 アプリケーションは **http://localhost:8080/** で使用できるようになります。



注記

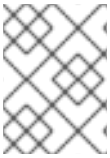
ROOT.war の名前を **jboss.war** に変更すると、アプリケーションは **http://localhost:8080/jboss** で使用できます。

1.6.2. ブートストラップの設定

マイクロコンテナブートストラップの設定は参照する **conf/bootstrap.xml** と **conf/bootstrap/*.xml** に記述されます。 今後、ブートストラップ Bean の数は削減される予定です。 通常のインストールでは、 ブートストラップ設定ファイルの変更は必要ないはずです。

1.6.3. レガシーコアサービス

サーバーがマイクロコンテナを開始した直後に、**conf/jboss-service.xml** ファイル内に指定されるレガシーコアサービスが開始されます。 このファイルをエディタで見ると、 ロギング、 セキュリティ、 JNDI、 JNDIView など各種サービスの MBean があることが分かります。 **JNDIView** サービスのエントリをコメントアウトしてみてください。



注記

サービスは、デプロイディレクトリサービスとしてデプロイされるマイクロコンテナ Bean か MBean に変換されるため、このファイルは最終的にドロップされます。

mbeans 定義はコメントはネストされたコメントがあるため、 mbean を 2 つのセクションでコメントアウトする必要があり、 オリジナルのコメントはそのまま残した点に注意してください。

```
<!-- 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>
-->
```

これで JBoss を再起動すると、 **JNDIView** サービスが JMX Management Console (JMX コンソール) の一覧に出現しなくなっていることがわかります。 実際には、 このファイルを修正する必要があることはめったにありませんが、 ファイルに追加の MBean エントリを追加したい場合は行っても構いません。 別の方法として、 **deploy** ディレクトリにある別のファイルを使用します。 これによりサービスがホットデプロイメント可能になります。

1.6.4. ロギングサービス

JBoss では、ロギングに **log4j** が使用されています。 **log4j** パッケージに関する知識がなく、アプリケーションに使用した場合は、Jakarta Web サイト (http://jakarta.apache.org/log4j/) で詳細を確認してください。

ロギングは中央の **conf/jboss-log4j.xml** ファイルより制御されます。 このファイルは、ログファイル、 ログファイルに記録されるメッセージのカテゴリ、 メッセージ形式、 フィルタのレベルを指定するアペンダのセットを定義します。 デフォルトでは、 JBoss はコンソールとログファイル (**log/server.log**) の両方に出力します。

TRACE、**DEBUG**、**INFO**、**WARN**、**ERROR**、**FATAL** の 6 つの基本ログレベルが使用されます。 コンソールのロギングしきい値は **INFO** で、コンソール上で情報メッセージ、 警告メッセージ、 エラーメッセージは表示されますが、 一般的なデバッグメッセージやトレースメッセージは表示されません。 これに対して **server.log** ファイルにはしきい値がないため、デフォルトは**DEBUG**に設定されています。

何らかの問題が発生し、コンソール内に有用な情報がないような場合は、 必ず**server.log** ファイルを確認し、問題の追跡に役立つデバックファイルがあるかチェックします。 ただし、 ロギングしきい値がデバッグメッセージの表示を許可しても、 JBoss 全体がログファイルに対して詳細なデバッグ情

報を生成するとは限りません。 また、 個別カテゴリのロギング限度を高く設定する必要があります。 例として次のカテゴリを見てください。

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

上記は、オーバーライドが特定されているクラスを除く全ての JBoss クラスに対してロギングレベルを **INFO** に制限します。 デフォルトでは、**jboss-log4j.xml** のルートロガーは **INFO** に設定されます。 そのため、全てのロガーカテゴリの **TRACE** ロガーや **DEBUG** ロガーはファイルやコンソールアペンダには記録されません。 この設定は jboss.server.log.threshold プロパティによって制御されます。 デフォルト設定は **INFO** になります。設定を **DEBUG** に変更すると、より詳細なログ出力が生成されます。 設定を変更する方法は 2 つあります。

- サーバー開始中に -Djboss.server.log.threshold=DEBUG パラメータを渡すことができます。

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

- ***JBOSS_DIST*/jboss-as/server/*PROFILE*/conf/jboss-log4j.xml** ファイルを直接編集して、このプロパティを設定することができます。

  ```
  <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>
  ```

注記

デフォルトでは、***JBOSS_DIST*/jboss-as/server/*PROFILE*/conf/jboss-log4j.xml** が 60 秒ごとにスキャンされ、変更がチェックされます。 このファイルへの変更後、60 秒以内に変更がホットデプロイされるため、ファイルを変更してもサーバーを再起動する必要はありません。

もう1 つの例として、 生成された SQL コマンドを解析するためにコンテナ管理の永続エンジンからの出力を **DEBUG** レベルに設定して別のファイル **cmp.log** にリダイレクトしたいとします。 この場合、**conf/jboss-log4j.xml** ファイルに以下のコードを追加することになります。

```
<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>
```

これにより、新ファイルの appender が作成され、 パッケージ **org.jboss.ejb.plugins.cmp** のロガー (またはカテゴリ) で使用されるように指定されます。

ファイルアペンダは、 サーバーが再起動される度に新しいファイルを生成したり、 1 つのファイルに永久に書き込みを行うのではなく、 毎日新しいログファイルを 1 つ作成するように設定されています。 現在のログファイルは **cmp.log** です。 古いファイルはファイル名に書き込みが行われた日付が追加されます。 また、 **log** ディレクトリにはウェブコンテナが作成する HTTP 要求のログも格納されます。

デフォルトでは、サーバーの再起動後、次のサーバー再起動までログメッセージを保持するよう **server.log** アペンダーが設定されています。 この動作は、**server.log** ファイルに対応する **FILE** アペンダーの Append プロパティによって制御されます。デフォルトでは、このプロパティは true に設定されています。 サーバーの再起動時に **server.log** の内容が削除されるようにするには、***JBOSS_DIST*/jboss-as/server/*PROFILE*/conf/jboss-log4j.xml** ファイルを編集し、このプロパティの値を false に設定します。 例は次の通りです。

```
<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. セキュリティサービス

セキュリティドメインの情報は named セキュリティドメインの一覧として **conf/login-config.xml** ファイル内に保存されます。 各 指定セキュリティドメインは、ドメイン内の認証目的で使用される JAAS [1] ログインモジュールを指定します。 アプリケーションでセキュリティを使用したい場合、 そのアプリケーションの JBoss 固有のデプロイメント記述子である **jboss.xml** (アプリケーションの JBoss 固有の設定定義に利用) や **jboss-web.xml** (Web アプリケーションに JBoss の定義に利用) に使用したいドメイン名を指定します。JBoss に同梱される JMX コンソールアプリケーションを保護する方法を簡単に見てみましょう。

JBoss サーバーのほぼ全体を JMX コンソールで制御することができます。 このため、 最低でもパスワードでアプリケーションを保護することが重要です。 パスワードで保護しないと、リモートユーザーがサーバーを自由に操作できてしまいます。 保護するには、 セキュリティドメインを追加してアプリケーションを保護の対象にします。 この作業は、 **deploy/jmx-console.war/WEB-INF/** ディレクトリにある JMX コンソールの **jboss-web.xml** ファイルで行います。 以下のように、ファイル内の **security-domain** をアンコメントします。

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

これがセキュリティドメインを Web アプリケーションにリンクします。 しかし、 Web アプリケーションに対して強化するセキュリティポリシー、 保護対象のURL、それらにアクセス権を持つユーザーなどは指示しません。 これを設定するには、 同じディレクトリ内の **web.xml** ファイルに行き、すでにそこにある **security-constraint** のコメントを外します。 このセキュリティ制約には **JBossAdmin** グループ内のユーザーの有効なユーザー名とパスワードが必要になります。

```
<!--
   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>
```

このユーザー名とパスワードはどこから来るのでしょうか？ アプリケーションをリンクした **jmx-console** セキュリティドメインから由来しています。 **conf/login-config.xml** にこの設定を行っています。

```
<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>
```

この設定はセキュリティポリシーを基にした簡単なファイルを使用します。 設定ファイルはサーバー設定の **conf/props** ディレクトリ内にあります。 ユーザー名とパスワードは**conf/props/jmx-console-users.properties** ファイルに保管され、 **"username=password"** の形式になっています。 ユーザーを **JBossAdmin** グループに割り当てるには、 **"username=JBossAdmin"** を **jmx-console-roles.properties** ファイルに追加します (ユーザー名の追加ロールはカンマで区切って追加できます)。 この既存のファイルはパスワードが **admin** となるユーザー **admin** を作成します。 セキュリティのために、 ユーザーを削除するかパスワードを安全性の高い別のパスワードに変更してください。

web.xml を更新すると必ず、 JBoss は JMX コンソールの再デプロイメントを行います。 サーバーコンソールをチェックして JBoss が変更を認識していることを確認することができます。 全てが正しく設定されアプリケーションの再デプロイメントが行われると、次に JMX コンソールにアクセスする時にユーザー名とパスワードが求められるようになります。[2]

JBoss に対する Web ベースの管理インターフェースは JMX コンソールだけではありません。 Web コンソールもその 1 つです。 これは Java アプレットですが、 該当の Web アプリケーションは JMX コンソールと同じ方法で保護する**deploy/management/console-mgr.sar/web-console.war.** にあり

ます。 唯一の違いは、 Web コンソールが JMX コンソールのように開かれたディレクトリ構成ではなく、簡単な WAR ファイルで提供されているという点です。機能上の違いは、WAR ファイル内でのファイル編集はやりにくいという点のみになります。

1.6.6. 追加サービス

コアではない、 ホットデプロイメントが可能なサービスは **deploy** ディレクトリに追加されます。 これらのサービスは、 XML 記述子ファイル、 **`*-service.xml`**、 **`*-jboss-beans.xml`**、 MC **`.beans`** アーカイブ、 JBoss Service Archive (SAR) ファイルのいずれになります。 SAR には、 **`META-INF/jboss-service.xml`** 記述子とサービスが必要とする追加リソース (例、 クラス、 ライブラリ JAR ファイル、 他のアーカイブなど) が含まれ、 すべてが単一のアーカイブにパッケージ化されます。 同様に、 **`.beans`** アーカイブには **`META-INF/jboss-beans.xml`** と追加リソースが含まれます。

これらのサービスに関する詳細は **JBoss Enterprise Application Platform: 管理設定ガイド** でご覧になれます。 また、 このガイドにはサーバーの内部機能及び JTA や J2EE Connector Architecture (JCA) などのサービス実装に関する総合的な情報も記載されています。

1.7. サービスバインディングマネージャー

JBoss サーバーは、提供するサービスに様々なポートを使用します (HTTP は 8080 番ポート、JNDI は 1099 番ポートなど)。 サービスバインディングマネージャー (SBM、Service Binding Manager) サービスは、ポートにバインドする必要がある全てのサービスを設定できる中央の場所を提供します。 SBM を使用すると、サーバーインスタンスに対して異なるポートバインディングのセットを設定できます。 SBM のシステムプロパティは、特定のサーバーインスタンスによって使用される named セット (ports-default、ports-01など) を制御します。 同じシステムで複数のサーバーインスタンスを実行するには、各インスタンスの SBM を設定し、異なる named バインディングセットを使用するようにします。 SBM を使用して、サーバーインスタンスに対して異なるバインディングセット (例：HTTP にデフォルトの 8080 番ポートではなく、8180番 ポートを割り当て) を割り当てることもできます。

一般的な設定では、**`ports-default`** セットは標準ポート (例: JNDI の場合は 1099 番ポート) を使用します。 **`ports-01`** は各ポートの値に 100 を追加 (例: JNDI の場合は 1199 番ポート)、 **`ports-02`** は 200 を追加というように指定されます。

SBM は ***`JBOSS_DIST`*`/jboss-as/server/`*`PROFILE`*`/conf/bindingservice.beans/ META-INF/bindings-jboss-beans.xml`** ファイルにより設定されます。 **`ServiceBindingManager`** の設定には主に 3 つの要素が関係します。

- 基準 (デフォルト) のバインディング設定データを含む Bean のセット。 **`ports-default`** や **`ports-01`** などを操作する基本値 (JNDI の場合 1099 番ポート) になります。
- **`ServiceBindingSets`** を定義する複数の Bean (例: **`ports-default`**、**`ports-01`**、**`ports-02`** など)。 基準のバインディングセットは、これらの各 Bean や、基準のポート値に適応されるオフセット値 (例: **`ports-01`** は 100) と組み合わされ、セットのバインディング値が生成されます。
- **`ServiceBindingManager`** サービス Bean 自体。基準のバインディングと投入する **`ServiceBindingSets`** を持っています。 サーバーインスタンスが使用すべきであるバインディングセットの名前で設定されます。 使用されるバインディングセットの名前は、jboss.service.binding.set システムプロパティを使用して、コマンドラインから設定できます。 デフォルト値は **`ports-default`** です。

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

```
   <constructor>

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

デフォルトで使用するポートセットから別のセットに切り替えるには、 **-Djboss.service.binding.set** プロパティを渡して下記の通りコマンドを実行し、サーバーを起動します。

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

これにより、**ports-01** バインディングセットに設定されているポートグループを使用するようサーバーが指示されます。

---

[1] Java Authentication and Authorization Service の略。 JBoss は JAAS を使用してプラグ可能な認証モジュールを提供します。 提供される JAAS を使用できますが、 特別な要件がある場合は独自の JAAS を作成することもできます。

[2] ユーザー名とパスワードは Web ブラウザのセッション変数であるため、ログインダイアログウィンドウを使用するにはブラウザーを再起動する必要がある場合があります。

第2章 他のデータベースの使用

JBoss Enterprise Application Platform には、設定済みのデータソースサンプルが同梱されており、同梱の Hypersonic データベースをすぐに利用することができます。このデータソースは、JNDI 名 **java:/DefaultDS** にバインドされており、この記述子は***JBOSS_DIST*/jboss-as//server/*PROFILE*/deploy/hsqldb-ds.xml** です。

DefaultDS JNDI 名と **hsqldb-ds.xml** 設定は通常のプラットフォームでの操作には必要ありません。本番用のインスタンスをデプロイする前にこのデータソースを削除するようにしてください。

警告

デフォルトの永続設定は Hypersonic (HSQLDB) に同梱されるため、JBoss Enterprise Platforms は「出荷された状態」で実行できます。ただし、**Hypersonic は本番稼働ではサポートされず、本番稼働環境では使用しないでください。**

Hypersonic Database の既知の問題は次のとおりです。

- トランザクション隔離がない
- スレッドおよびソケットのリーク (**connection.close()** がリソースを整理しない)
- 永続品質 (障害発生後にログが破損し、自動回復が行えない)
- データベースの破損
- ロード状態での安定性 (扱うデータが大きすぎると、データベースプロセスが消失する)
- クラスター環境で実行不可

この章では、JBoss Enterprise Application Platform を今日最も使用されているデータベースサーバーに接続するためのデータソースの設定方法やデプロイ方法を詳しく説明します。

2.1. DATASOURCE の設定ファイル

DataSource 設定ファイルの名前は接尾辞の **-ds.xml** で終わっているため、 JCA デプロイヤーで正しく判別されるようになっています。 **docs/example/jca** ディレクトリには、 様々な種類のデータベースのサンプルファイルが格納されており、これらを出発点として使用するといいでしょう。 設定フォーマットの総合的な説明については、DTD ファイル **docs/dtd/jboss-ds_1_5.dtd** を確認するとよいでしょう。 これらのファイルや JBoss JCA 実装についての追加のドキュメントは、http://www.redhat.com/docs/en-US/JBoss_Enterprise_Application_Platform/ にある JBoss Enterprise Application Platform の 管理およびサーバー設定ガイドを参照してください。

ローカルトランザクションデータソースは **local-tx-datasource** 要素を使って設定され、XA 準拠のデータソースは、**xa-datasource** を使って設定されます。 サンプルファイル**generic-ds.xml** は、両タイプの使用法や、接続プール設定などに使用できるその他要素の使用法について説明しています。 ローカル設定のサンプルと XA 設定のサンプルは、 Oracle、DB2、Informix で使用できます。

サンプルファイル **firebird-ds.xml**、 **facets-ds.xml**、 **sap3-ds.xml** を見ると、全く異なる形式であることが分かるはずです。root エレメントは **datasources** ではなく、**connection-factories** になっています。 これらのファイルは、事前にパッケージされた JCA リソースアダプターで使用される、より汎用的な別の JCA 設定構文を使用します。 この構文はデーターソース設定固有の構文ではなく、 JMS リソースアダプタを設定するために ***JBOSS_DIST*/jboss-as/server/*PROFILE*/deploy/messaging/jms-ds.xml** ファイル内でも使用されます。

次に例の手順を一つずつ追って、特定の データベース用の datasource のセットアップに 必要な事項を説明していきます。

2.2. MYSQL をデフォルト DATASOURCE として使用

MySQL® は、一貫した高速パフォーマンスや高可用性、使いやすさから、世界で最も使用されているオープンソースのデータベースとなりました。 このデータベースサーバーは、大企業から専用の組み込みアプリケーションまで、世界各地で多くこの設定が使用されています。 公式な JDBC ドライバーは **Connector/J** と呼ばれます。 この例では、MySQL 5.1.31 と **Connector/J** 5.1.8 を使用しています。 これらは両方、http://www.mysql.com よりダウンロードできます。

2.2.1. データベースとユーザーの作成

ユーザーは MySQL の基本について知っていて、既に MySQL をインストール済みで、実行中であることを仮定します。 コマンドラインから **mysql** クライアントプログラムを実行し、管理コマンドを実行できるようにします。 十分な権限を持ったユーザーとして接続するようにしてください (たとえば、MySQL root ユーザーとして実行するには、 **-u** root オプションを指定します)。

最初に、JBoss が使用する **jboss** というデータベースを MySQL 内に作成します。

```
mysql> CREATE DATABASE jboss;

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

データベースが作成されたことを確認します。

```
mysql> SHOW DATABASES;

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

次に、**jboss** というユーザーを作成し、データベースへアクセスするためのパスワードを password にします。

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

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

ユーザーとパスワードが作成されたことを確認します。

```
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. JDBC ドライバーのインストールとデータソースの導入

JDBC ドライバークラスを JBoss Enterprise Application Platform で使用できるようにするには、**Connector/J** ディストリビューションのアーカイブ **mysql-connector-java-5.1.8-bin.jar** を、**default** サーバー 設定内の **lib** ディレクトリにコピーします (サーバー設定が実行されていることを仮定します)。

以下のデータソース設定で、**mysql-ds.xml** というファイルを deploy ディレクトリ内に作成します。データーベースユーザー名とパスワードは、前の項で作成した MySQL ユーザーと同様です。

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
   <local-tx-datasource>
      <jndi-name>DefaultDS</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>
      <metadata>
         <type-mapping>mySQL</type-mapping>
      </metadata>
      <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-
isolation>
   </local-tx-datasource>
</datasources>
```

***JBOSS_DIST*/jboss-as/server/*PROFILE*/deploy** フォルダーに正しくデータソースが設定されたことを確認するには、サーバーを起動し、次のようなメッセージがログにあるか確認します。

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

注記

他のデータソースの設定手順も同様です。

2.2.3. MySQL DataSource のテスト

「JDBC クライアントの作成」に説明のあるテストクライアントを使用し、データソースが適切にインストールされているか検証できます。

2.3. ORACLE DB に対するデータソースの設定

Oracle は商用データベースの分野では主要企業の 1 つで、多くのユーザーが Oracle に馴染みがあるはずです。 非商用の目的であれば、http://www.oracle.com/technology/products/database/xe/index.html より無料でダウンロードすることができます。

本項では、 http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/index.html にある最新の JDBC ドライバ (11g) を使用して、サーバーを Oracle Database 11g Express Edition に接続します。

2.3.1. JDBC ドライバーのインストールとデータソースの導入

JDBC ドライバクラスを JBoss Enterprise Application Platform で使用できるようにするには、 アーカイブ **ojdbc6.jar** をデフォルトサーバー 設定内の **lib** ディレクトリにコピーします (サーバー設定が実行されていることを仮定します)。

以下のデータソース記述子を用いて、**oracle-ds.xml** というテキストファイルを **deploy** ディレクトリに作成します。

```
<?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>
```

データソースは、Oracle XE でデフォルトで提供される **xe** というデータベース/SID を示しています。

環境設定に合わせるため、接続 url 属性と、ユーザー名とパスワードの組み合わせをアップデートする必要があります。

2.3.2. Oracle DataSource のテスト

Oracle XE と JBoss Engerprise Application Platform は、共にデフォルトで Webサーバーを 8080 番ポートで起動するため、ポート競合を防ぐために Oracle XE を再設定してから、データソース設定を検証する必要があります。

Oracle SQL コマンドラインを開き、次のコマンドを実行します。

```
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
```

これで、http ベースの管理ツールを提供するために Oracle XE によって起動される Web サーバーが 8090 番ポートで実行されます。 通常通りに JBoss Enterprise Application Platform サーバーインスタンスを開始します。 これで、テストクライアントを使用してデータソースの適切にインストールされているか検証できるようになります。

2.4. MICROSOFT SQL SERVER 200X に対するデータソースの設定

本項では、http://msdn2.microsoft.com/en-us/data/aa937724.aspx にある最新の JDBC ドライバ (v2.0) を使用して、サーバーを MS SQL Server 2005 に接続します。

2.4.1. JDBC ドライバーのインストールとデータソースの導入

JDBC ドライバクラスを JBoss Enterprise Application Platform で使用できるようにするには、**sqljdbc_2.0** ディストリビューションのアーカイブ **sqljdbc.jar** を、デフォルトサーバー設定内の **lib** ディレクトリにコピーします (サーバー設定が実行されていることを仮定します)。

以下のデータソース記述子を用いて、**mssql-ds.xml** というテキストファイルを **deploy** ディレクトリに作成します。

```
<?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>
```

データソースは、MS SQL Server 2000 でデフォルトで提供される **pubs** というデータベースを示しています。

環境設定に合わせるため、接続 url 属性と、ユーザー名とパスワードの組み合わせをアップデートするようにしてください。

2.4.1.1. データソースのテスト

「JDBC クライアントの作成」に説明のあるテストクライアントを使用し、データソースが適切にインストールされているか検証できます。

2.5. JBOSS MESSAGING 永続マネージャーの設定

JBoss Messaging の永続マネージャーはデフォルトのデータソースを使用してテーブルを作成し、メッセージやトランザクションデータ、その他インデックスを保存します。「永続」の設定は、**xxx-persistence-service.xml** ファイルにグループ化されます。 JBoss Enterprise Application Platform は、デフォルトの **hsqldb-persistence-service.xml** ファイルを標準装備しています。このファイルは、JBoss Enterprise application Platform にデフォルトで標準装備される Hypersonic データベースインスタンスを使用するよう Messaging サーバーを設定します。

all または **default** 設定を基にした設定内で **hsqldb-persistence-service.xml** ファイルを確認することができます。

- ***JBOSS_DIST*/jboss-as/server/all/deploy/hsqldb-persistence-service.xml**
- *JBOSS_DIST/jboss-as*/**server/default/deploy/hsqldb-persistence-service.xml**

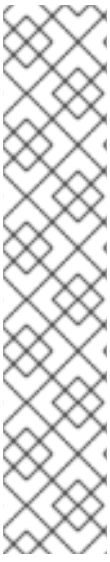
警告

Hypersonic データベースは、トランザクション分離のサポートに限りがあり、高負荷の状態での信頼性が低いため、実稼働環境での使用は推奨されません。

JBoss Messaging の設定に関する詳細は、『管理設定ガイド』 を参照してください。

代わりのメッセージング永続設定

RPMからインストールした場合、**/usr/share/doc/jbossas-5.1.0/examples/jms/** にある代わりのメッセージング永続設定ファイルを、ZIPアーカイブあるいは RPM からインストールした場合は***JBOSS_DIST*/jboss-as/docs/examples/jms/** の設定ファイルを参照できます。



注記

正しいデータソースファイルをデプロイした後、古いデータソースを削除します。

- ***JBOSS_DIST*/*jboss-as*/server/*PROFILE*/deploy/hsqldb-ds.xml**

クラスタ設定に対してメッセージング永続設定をアップデートするには、このファイルを編集します。

- ***JBOSS_DIST*/*jboss-as*/server/*PROFILE*/deploy/messaging/hsqldb-persistence-service.xml**

クラスタ化属性を Trueに設定する方法：

```
<attribute name="Clustered">true</attribute>
```

2.6. JDBC クライアントの作成

新しく設定されたデータソースをテストする場合は、JSP ページに組み込まれた非常に基本的な JDBC クライアントコードを使用することを推奨します。最初に、**"jdbcclient.war"** という名前のフォルダである展開された WAR アーカイブを deploy ディレクトリ下に作成する必要があります。このフォルダで、client.jsp という名前のテキストドキュメントを作成し、以下のコードを貼り付けます。

```
 <%@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();
 }
 } %>
```

Web ブラウザを開いて、http://localhost:8080/jdbcclient/client.jsp にアクセスします。JDBC クエリの結果としてユーザーのリストとパスワードが表示されるはずです。

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

付録A 改訂履歴

改訂 5.1.2-3.400 **2013-10-30** **Rüdiger Landmann**
Rebuild with publican 4.0.0

改訂 5.1.2-3 **2012-07-18** **Anthony Towns**
Rebuild for Publican 3.0

改訂 5.1.2-100 **Thu Dec 8 2011** **Jared Morgan**
JBoss Enterprise Application Platform 5.1.2 GAに対する変更を追加。本ガイド文書の変更に関する情報は、『リリースノート 5.1.2』を参照してください。

改訂 5.1.1-100 **Mon Jul 18 2011** **Jared Morgan**
JBoss Enterprise Application Platform 5.1.1 GAに対する変更を追加。本ガイド文書の変更に関する情報は、『リリースノート 5.1.1』を参照してください。

改訂 5.1.0-100 **Wed Sep 15 2010** **Laura Bailey**
新しいバージョン管理要件に応じてバージョン番号が変更されました。
JBoss Enterprise Application Platform 5.1.0.GA のために改訂されました。