



USING ENTERPRISE JAVABEANS 3.0 AND THE JAVA PERSISTENCE API WITH JBOSS AND PROGRESS® DATADIRECT CONNECT® *for* JDBC

OVERVIEW

This article provides all the information you need to use Progress DataDirect's JDBC drivers with these Java technologies. Java Platform, Enterprise Edition (Java EE) is the industry standard for developing portable, robust, scalable and secure server-side Java applications. Java EE provides web services, component model, management, and communications APIs that make it the industry standard for implementing enterprise-class service-oriented architecture (SOA) and next-generation web applications. The Enterprise JavaBeans (EJB) architecture is a component-based architecture for development and deployment of component-based distributed applications. Applications written using Enterprise JavaBeans are scalable, transactional, and multi-user secure. These applications may be written once, and then deployed on any server platform that supports the EJB specification. The Java Persistence API (JPA) provides a POJO (plain old Java Object) persistence model for object-relational mapping. It contains a full object/relational mapping specification supporting the use of Java language metadata annotations and/or XML descriptors to define the mapping between Java objects and a relational database. It supports a rich, SQL-like query language (which is a significant extension upon EJB Query Language) for both static and dynamic queries. It also supports the use of pluggable persistence providers.

The JPA specification was introduced by Sun as a means to combine the best ideas from many sources in the new persistence API and create a lightweight, easy to use API which is integrated into any Java EE 5 application server.

JPA provides a standardized means for mapping relational data to Java objects, while also handling the way these objects are stored in the database and removing the need to code directly to the JDBC API.

Why Using DataDirect JDBC Drivers Makes This Technology Better

- ▶ Performance and scalability
 - > Best JDBC driver performance for any production scenario
 - > Best JDBC performance for single-threaded (one connection) and multi-threaded (many connections) situations

HIGHLIGHTS:

- ▶ Implement best JDBC performance drivers using these step-by-step instructions
- ▶ Connect drivers within JBoss and EJB
- ▶ Quickly add JPA support

- ▶ Quality and support
 - > Quality tested with extensive internal and industry test suites
 - > Embedded in the world's most demanding software applications and application servers
 - > Backed by the industry's best technical support organization with a complete focus on data access middleware
- ▶ Consistent and complete database support in one package
 - > Robust database feature support including security features such as SSL and Kerberos
 - > Supports the latest versions of all major databases
 - > Standards-based, 100% Java approach to feature implementation across databases

REQUIRED COMPONENTS

- ▶ Download JBoss 4.2.2 GA from http://sourceforge.net/project/showfiles.php?group_id=22866&package_id=16942&release_id=548923
- ▶ Download DataDirect Connect *for* JDBC 3.7 from http://www.datadirect.com/downloads/registration/connect_jdbc/index.ssp

INSTALL DATADIRECT CONNECT *for* JDBC 3.7

- ▶ Install the DataDirect Connect *for* JDBC drivers using the instructions in the [DataDirect Connect *for* JDBC Installation Guide](#).
- ▶ Verify connectivity to your database server using the steps detailed in the [Quick Start Guides](#).

CONFIGURATION / SETUP FOR DATADIRECT DRIVERS WITHIN JBOSS

Copy the DataDirect Connect *for* JDBC jar files [util.jar, base.jar, db2.jar, informix.jar, oracle.jar, sqlserver.jar, and sybase.jar] from:
install_dir/lib

to:

JBoss_home/server/server_config/lib

where *install_dir* is your DataDirect Connect *for* JDBC installation directory, *JBoss_home* is your JBoss installation directory, and *server_config* is your server configuration directory.

For example, to configure the DataDirect Connect *for* JDBC drivers for the default JBoss server configuration, copy the driver jar files to *JBoss_home/server/default/lib*.

HOW TO CONNECT

1. In the *JBoss_home/server/server_config/deploy* directory, create a data source file named *datadirect-ds.xml*. The file name must end with the characters *ds.xml* so that the JBoss server can recognize it as a data source file; however, the file name can be prefixed with any set of characters.
2. Edit the file as shown in the following example, adding or modifying the required XML tags as described in this step. The example shown is contained in the file *datadirect-ds.xml*, which is available from the same location you obtained this document.
 - > The value of the `<jndi-name>` tag specifies the JNDI name, which is used to look up the data source. In the following example, the JNDI name used to look up the data source is *ds/TestDS*. JBoss maps the driver to the global space. An application can look up the data source using the string:
`java:ds/TestDS`
 - > Modify the value of the `<driver-class>` tag to specify the appropriate class name for the driver to use. For example, the following code specifies the class name of the DataDirect Connect *for* JDBC SQL Server driver.
`<driver-class>`
`com.ddtek.jdbc.sqlserver.SQLServerDriver`
`</driver-class>`

- > Modify the value of the `<connection-url>` tag to specify the correct connection information for the driver and database server to use. For example, the following code specifies connection information used by the DataDirect Connect *for* JDBC SQL Server driver to connect to the server named myserver on port 1433.

```
<connection-url>
  jdbc:datadirect:sqlserver://myserver:1433
</connection-url>
```

- > Modify the value of the `<user-name>` and `<password>` tags to specify a valid user name and password for the database server. For example, the following code specifies the user name test and the password secret:

```
<user-name>test</user-name>
<password>secret</password>
```

- > The `<connection-property>` tag specifies DataDirect Connect *for* JDBC driver-specific connection properties. For example, the following code sets the value of the DataDirect Connect *for* JDBC SQL Server driver connection property `SendStringParametersAsUnicode` to false.

```
  <connection-property name="sendStringParametersAsUnicode">
false
</connection-property>
```

- > The tags following the `<!--pooling parameters-->` comment in the following example are properties that control JBoss connection pooling. Refer to the JBoss documentation for details on setting these properties. `<?xml version="1.0" encoding="UTF-8"?>`

```
<!-- ===== -->
<!-- -->
<!--   DataDirect Data Sources -->
<!-- -->
<!-- ===== -->
<!--
    See the generic _ds.xml file in the doc/examples/jca folder
    for examples of properties and other tags you can specify
    in data sources
-->
```

```
<datasources>
```

```
  <!-- JBossTest Data Source -->
  <local-tx-datasource>
    <jndi-name>ds/TestDS</jndi-name>
    <connection-url>
      jdbc:datadirect:sqlserver://myserver:1433
    </connection-url>
    <driver-class>
      com.ddtek.jdbc.sqlserver.SQLServerDriver
    </driver-class>

    <user-name>test</user-name>
    <password>secret</password>

    <!-- Driver Specific Options -->
    <connection-property name="sendStringParametersAsUnicode">
      false
    </connection-property>
```

```

        <!--pooling parameters-->
        <min-pool-size>5</min-pool-size>
        <max-pool-size>100</max-pool-size>

        <blocking-timeout-millis>5000</blocking-timeout-millis>
        <idle-timeout-minutes>15</idle-timeout-minutes>
    </local-tx-datasource>
</datasources>

```

3. **Generate the JSP Test Page** You must create a JSP page that uses the data source you created in the previous step. A sample JSP page named `JBossTest.jsp` is provided in the `JBossTestWeb.war` file, which is available from the same location you obtained this document. This JSP page includes the following code to look up the data source and obtain a connection to the database from the data source.

```

InitialContext ctxt = new InitialContext();
DataSource ds = (DataSource) ctxt.lookup("java:ds/TestDS");
con = ds.getConnection();

```

4. **Deploy the JBossTest Web Application to the JBoss Application Server**

Copy the `JBossTestWeb.war` file, available with this document, to the directory `JBoss_home/server/server_config/deploy`.

5. **Run the JBossTest Web Application** Start the JBoss application server by running `run.bat` or `run.sh`, located in the `JBoss_home/bin` directory. To start a configuration other than the default configuration, use the `-c` option when executing the script. For example to run the all configuration, use the following command:

```
run -c all
```

6. **Open a web browser and enter the following URL to display the JBossTest web page:**

`http://localhost:8080/JBossTestWeb/JBossTest.jsp`

The contents of the page display the version information of the driver and the database server to which it connects, if the DataDirect Connect *for JDBC* driver has been installed and configured correctly. For example, the following figure shows version information for the DataDirect Connect *for JDBC* SQL Server driver connecting to a database server running Microsoft SQL Server 2000.

7. **Map the Data Source to a Local JNDI Name in a Session EJB**

Typically an EJB does not use the global JNDI name to look up the data source. Instead, it uses a logical JNDI name that is mapped to the global JNDI name of the data source. To map a logical JNDI name to the global JNDI data source name in a Session EJB, declare a resource reference in the JBoss-specific deployment descriptor file `jboss.xml`. A resource reference is defined by adding the tag `<resource-ref>` as a child of the `<session>` tag as shown in the following example.

```

<session>
    <ejb-name>SupportedDatabases</ejb-name>

    <jndi-name>SupportedDatabasesBean</jndi-name>
    <local-jndi-name>SupportedDatabasesLocal</local-jndi-name>
    <resource-ref>
        <res-ref-name>jdbc/TestDS</res-ref-name>
        <jndi-name>java:/ds/TestDS</jndi-name>
    </resource-ref>
</session>

```

The value of the `<res-ref-name>` tag is the logical JNDI name an EJB uses to look up the data source. The value of the `<jndi-name>` tag is the global name of the data source to which the logical name is mapped.

Refer to your JBoss documentation for more information about JBoss-specific deployment descriptors.

8. Specify the Data Source for an Entity EJB

To specify the data source to be used with an Entity EJB, specify the global JNDI name of the data source in the JBoss-specific Container Manager Persistence [CMP] deployment descriptor, `jbosscmp-jdbc.xml`. The value of the `<datasource>` tag specifies the global JNDI name of the data source to use with Entity EJBs. The `<datasource>` tag can be specified as a child of the `<defaults>` tag as shown in the following example, or it can be specified as a child of a particular `<entity>` tag.

```
<defaults>
  <datasource>java:/ds/TestDS</datasource>
  <datasource-mapping>MS_SQLSERVER</datasource-mapping>
</defaults>
```

Refer to your JBoss documentation for more information about JBoss-specific deployment descriptors.

HOW TO ENABLE THE CONNECTION TO USE JPA

Adding support for JPA is achieved by defining the `persistence.xml` file in the META-INF directory within your application package. The `persistence.xml` file is the standard configuration file in JPA. In this file you define a uniquely named persistence unit which will be used by the `EntityManager` defined in your Entity Bean code. The provider attribute specifies the underlying implementation of the JPA `EntityManager`. In JBoss AS, the default and only supported / recommended JPA provider is Hibernate. The `jta-data-source` points to the JNDI name of the database connection to which this persistence unit maps.

For more details on the `persistence.xml` file, please refer to Chapter 6 of the [EJB 3.0 Java Persistence API Specification](#).

Sample `persistence.xml` file for this example:

```
<persistence>
  <persistence-unit name="TestDD">
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>java:/ds/TestDS</jta-data-source>
    <properties>
      <property name="hibernate.dialect"
        value="org.hibernate.dialect.SQLServerDialect"/>
    </properties>
  </persistence-unit>
</persistence>
```

USEFUL LINKS

JSR 220: Enterprise JavaBeans 3.0 Specification download: <http://jcp.org/aboutJava/communityprocess/final/jsr220/index.html>

JPA JavaDoc: <https://glassfish.java.net/nonav/javaee5/api/index.html?javax/persistence/package-summary.html>

Sun's Java Persistence Example [Uses NetBeans]: <https://glassfish.java.net/javaee5/persistence/persistence-example.html>

We welcome your feedback! Please send any comments concerning documentation, including suggestions for other topics that you would like to see, to: docgroup@datadirect.com

PROGRESS SOFTWARE

Progress Software Corporation [NASDAQ: PRGS] is a global software company that simplifies the development, deployment and management of business applications on-premise or in the cloud, on any platform or device, to any data source, with enhanced performance, minimal IT complexity and low total cost of ownership.

WORLDWIDE HEADQUARTERS

Progress Software Corporation, 14 Oak Park, Bedford, MA 01730 USA Tel: +1 781 280-4000 Fax: +1 781 280-4095 On the Web at: www.progress.com

Find us on  facebook.com/progresssw  twitter.com/progresssw  youtube.com/progresssw

For regional international office locations and contact information, please go to www.progress.com/worldwide

Progress DataDirect, DataDirect Connect, and SequeLink are trademarks or registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and other countries. Any other marks contained herein may be trademarks of their respective owners. Specifications subject to change without notice.

© 2008, 2014 Progress Software Corporation. All rights reserved.

Rev. 9/14

www.progress.com

