Composer Help

Connection Pooling

4/3/2025

Connection Pooling

Contents

When defining a database connection profile, you can use connection pooling, which maintains a set of database connections that can be reused for requests to databases. This feature can enhance performance by avoiding time-consuming re-establishment of connections to databases. While Composer does not support specific application servers, this topic presents information on configuring Tomcat, JBoss, and Websphere application servers to expose a pooled data source as a JNDI resource. This topic also contains information on creating a JDBC provider for an Oracle database.

> **Important**
>
> When upgrading projects to a 8.1.5 branch (starting with version 8.1.500.03 to 8.1.541.07), please perform the following steps to avoid issues with DB connection pooling:
>
> 1. Stop the Tomcat service (if it happens to be a service) or manually shut down the server.
> 2. Download the `mchange-commons-java-0.2.15.jar` file from the Maven repository (https://mvnrepository.com/artifact/com.mchange/mchange-commons-java/0.2.15).
> 3. Copy the downloaded JAR to the <tomcat_installed_directory>\lib location in the physical directory.
> 4. Restart Tomcat.

Note: You do not have to perform the above steps if you are upgrading to a 8.1.5 branch that was released after version 8.1.541.07.

Connection Pooling for Tomcat Application Servers

For Tomcat, a JNDI resource is defined in a Context configuration. Do this in the global scope, at `$TOMCAT_HOME/conf/context.xml`. Here is a sample:

```
 <Context> ...

<Resource name="jdbc/pooledDS" auth="Container"
        type="com.mchange.v2.c3p0.ComboPooledDataSource"
        factory="org.apache.naming.factory.BeanFactory"
        driverClass="com.microsoft.sqlserver.jdbc.SQLServerDriver"
        user="john"  password="doe123"
        jdbcUrl="jdbc:sqlserver://dbserver1:1433;databaseName=composer1" />

<Resource name="jdbc/oraclePooled" auth="Container"
        type="com.mchange.v2.c3p0.ComboPooledDataSource"
        factory="org.apache.naming.factory.BeanFactory"
        driverClass="oracle.jdbc.driver.OracleDriver"user="jane"password="doe456"
```

```
jdbcUrl="jdbc:oracle:thin:@dbserver2:1521:composer2" />... </Context>
```

Important Items

- `name`--should match the Connection Pool Name parameter given in the Connection Profile in Composer.
- `user`, `password`--these are the login credentials to the database.
- `jdbcUrl`--specifies the host, port and database name. Can be copied from the Connection Profile editor in Composer. The JDBC URL can also use advanced options that might not be otherwise exposed by Composer. For example, to enable Transparent Application Failover for a connection to an Oracle database, the URL can be given as:

```
jdbcUrl="jdbc:oracle:oci:@(DESCRIPTION=(LOAD_BALANCE=on)(FAILOVER=on)(ADDRESS=(PROTOCOL=tcp)(HOS
(ADDRESS=(PROTOCOL=tcp)(HOST=host2)(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=dbcluster)
(FAILOVER_MODE=(TYPE=session)(METHOD=basic))))"
```

Additional Pooling Parameters

Additional pooling parameters can be customized here as well, for example:

```
<Resource name="jdbc/pooledDS" auth="Container"
type="com.mchange.v2.c3p0.ComboPooledDataSource"
factory="org.apache.naming.factory.BeanFactory"
driverClass="com.microsoft.sqlserver.jdbc.SQLServerDriver"
user="john"
password="doe123"
jdbcUrl="jdbc:sqlserver://dbserver1:1433;
databaseName=composer1"
maxPoolSize="20" acquireRetryAttempts="0" /
```

For a full list of available settings, refer to the c3p0 documentation, which is the third-party connection pooling library used by Composer

http://www.mchange.com/projects/c3p0/index.html
http://www.mchange.com/projects/c3p0/index.html.

Connection Pooling for JBoss Application Servers

To define connection pooling for JBoss:

1. Add the c3p0 and JDBC driver JARs to JBoss's `global lib` directory (`$JBOSS_HOME/server/<instance>/lib`). This is because JBoss will initialize the connection pool upon startup regardless of what applications are deployed. This is in contrast to Tomcat, which creates the connections on demand.
2. Next, define the JNDI resources in a file called `c3p0-service.xml`. Copy the file into `$JBOSS_HOME/server/<instance>/deploy`.

Sample:

```
 <?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE server> <server> <mbean
code="com.mchange.v2.c3p0.jboss.C3P0PooledDataSource"
name="jboss:server=SQLServerDS">     <attribute name="JndiName">java:jdbc/
pooledDS</attribute>     <attribute
name="JdbcUrl">jdbc:sqlserver://dbserver1:1433;databaseName=composer1</attribute>
     <attribute
name="DriverClass">com.microsoft.sqlserver.jdbc.SQLServerDriver</attribute>
     <attribute name="User">john</attribute>     <attribute
name="Password">doe123</attribute> </mbean> <mbean
code="com.mchange.v2.c3p0.jboss.C3P0PooledDataSource" name="jboss:server=OracleDS">
     <attribute name="JndiName">java:jdbc/oraclePooled</attribute>     <attribute
name="JdbcUrl">jdbc:oracle:thin:@dbserver2:1521:Composer2</attribute>     <attribute
name="DriverClass">oracle.jdbc.driver.OracleDriver</attribute>     <attribute
name="User">jane</attribute>     <attribute name="Password">doe456</attribute>
</mbean> </server>
```

Pooling Parameters

Specify pooling parameters are specified by adding more <attribute> elements, e.g.,

```
<mbean code="com.mchange.v2.c3p0.jboss.C3P0PooledDataSource"
name="jboss:server=OracleDS">     <attribute name="JndiName">java:jdbc/
oraclePooled</attribute>     <attribute name="JdbcUrl">jdbc:oracle:thin:@dev
dbserver2:1521:Composer2</attribute>     <attribute
name="DriverClass">oracle.jdbc.driver.OracleDriver</attribute>     <attribute
name="User">jane</attribute>     <attribute name="Password">doe456</attribute>
     <!-- note that the attribute names must be capitalized -->     <attribute
name="MaxPoolSize">20</attribute>     <attribute
name="AcquireRetryAttempts">0</attribute> </mbean> For a full list of available settings, refer to
the c3p0 documentation, which is the third-party connection pooling library used by Composer
([http://www.mchange.com/projects/c3p0/index.html http://www.mchange.com/projects/c3p0/
index.html]).
```

Configuration Files

The following configuration files are automatically generated by Composer's WAR export functionality and do not require any user action: web.xml and jboss-web.xml

web.xml In the web application itself, the deployment descriptor (`WEB-INF/web.xml`) needs to specify a resource reference: `<resource-ref> <res-ref-name>jdbc/pooledDS</res-ref-name><res-type>javax.sql.DataSource</res-type><res-auth>Container</res-auth></resource-ref>` **jboss-web.xml**

This special JBoss-specific configuration file (WEB-INF/jboss-web.xml) is required to map the resource-ref to the globally defined resource.

```
<?xml version="1.0" encoding="UTF-8"?> <jboss-web> <resource-ref> <res-ref-name>jdbc/
pooledDS</res-ref-name><res-type>javax.sql.DataSource</res-type><jndi-name>java:jdbc/
pooledDS</jndi-name></resource-ref> </jboss-web>
```

Connection Pooling for WebLogic Application Servers

When the application Server is WebLogic, there must be an extra configuration file in WEB-INF called `weblogic.xml`. First, though, confirm that the following is present in `web.xml` in the exported .war file:

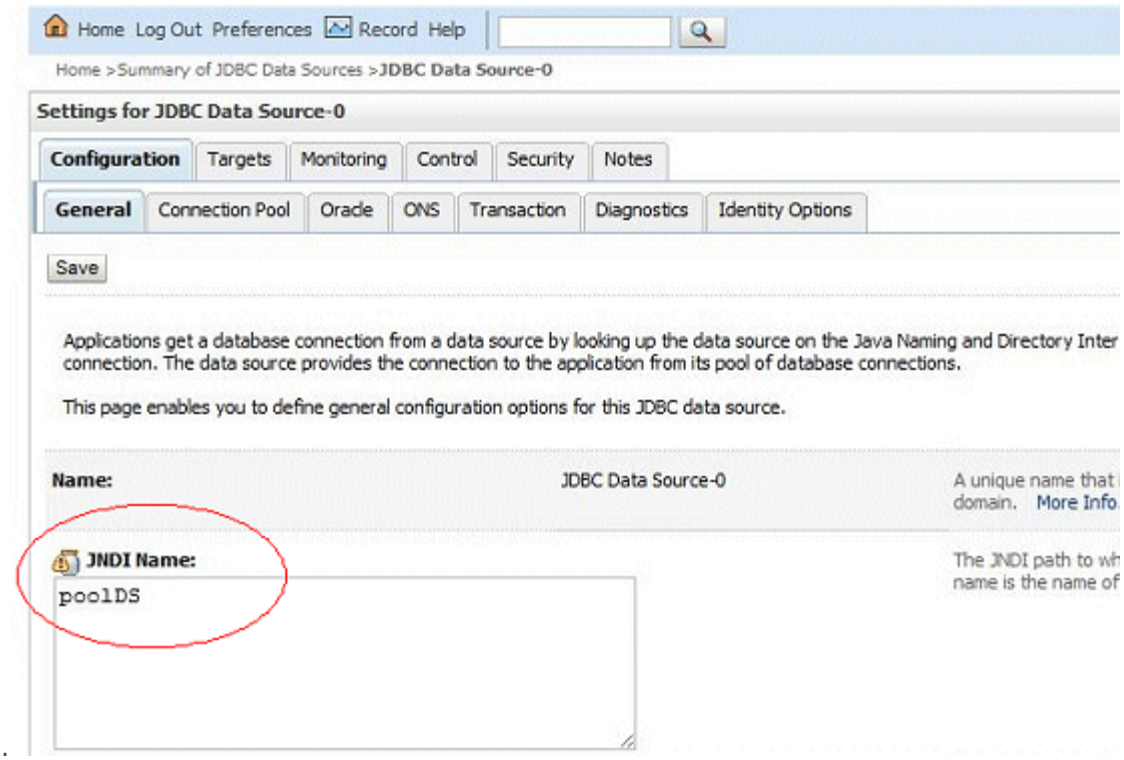
`<resource-ref> res-ref-name>jdbc/poolDS</res-ref-name> </res-type>`
`<res-auth>Container</res-auth> </resource-ref>`
`res-ref-name` should match the pool name in the connection.properties file, and it should be prefixed by `jdbc/`

weblogic.xml File The `weblogic.xml` can be added to the Composer Project in WEB-INF. Afterwards, you will have to export the .war file from Composer again and redeploy. The `weblogic.xml` should contain:

```
<?xml version="1.0" encoding="UTF-8"?>
<wls:weblogic-web-app xmlns:wls="http://xmlns.oracle.com/weblogic/weblogic-web-app" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/ejb-jar_3_0.xsd http://xmlns.oracle.com/weblogic/weblogic-web-app http://xmlns.oracle.com/weblogic/weblogic-web-app/1.2/weblogic-web-app.xsd">
<wls:weblogic-version>12.2.1.3.0</wls:weblogic-version>
<wls:context-root>JavaComposerProject</wls:context-root>
<wls:resource-description>
<wls:res-ref-name>jdbc/poolDS</wls:res-ref-name>
</wls:resource-description>
</wls:weblogic-web-app>
```

Note the following:

The `wls:res-ref-name` should match `res-ref-name` in `web.xml`. `wls:jndi-name` should be the JNDI Name in the WebLogic configuration.



Connection Pooling for WebSphere Application Servers

WebSphere has its own connection pooling capabilities, so you won't be using c3p0. The data sources are defined in the WebSphere management console.

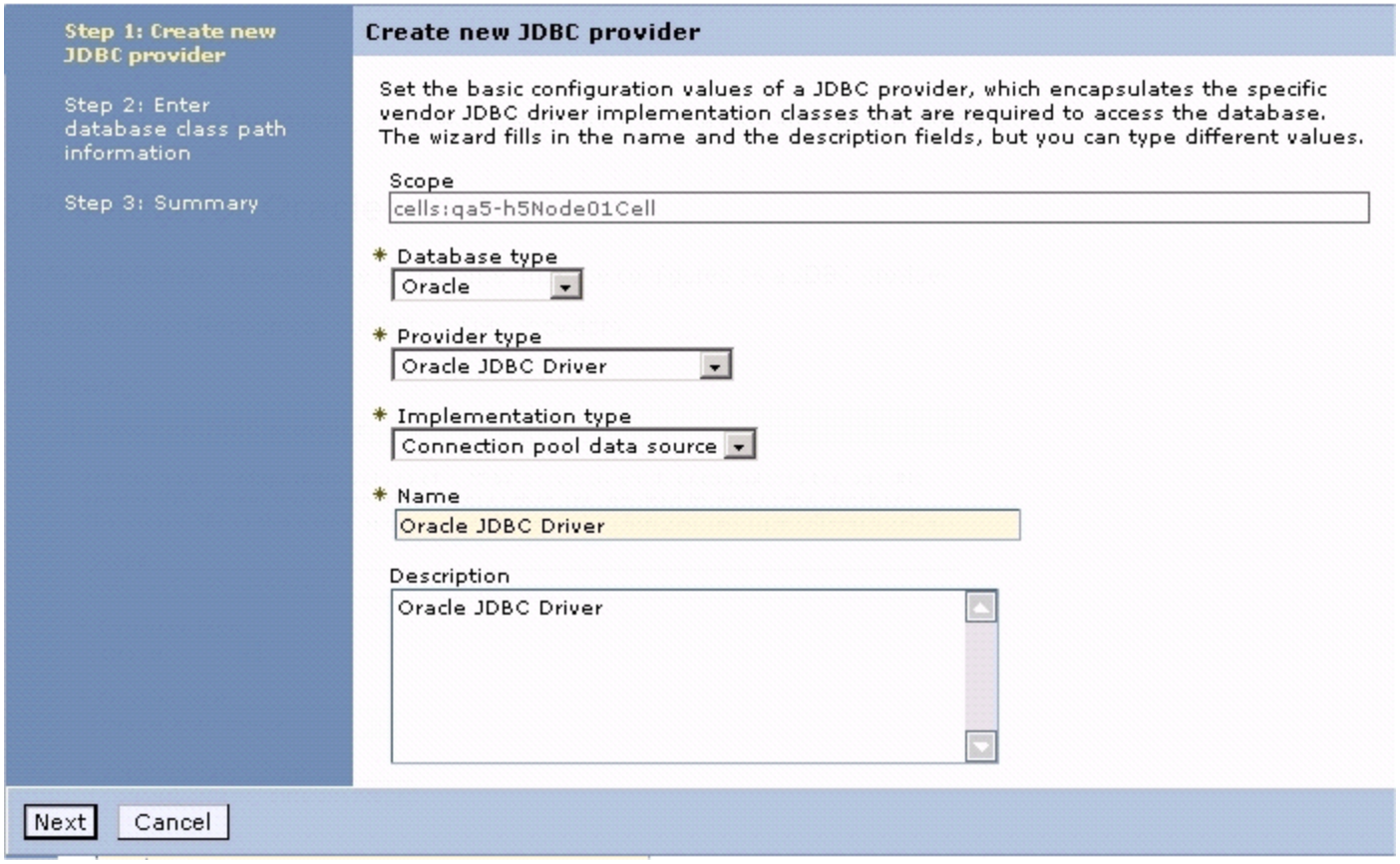
Configuration Files

The following configuration files are automatically generated by Composer's WAR export functionality and do not require any user action: web.xml and ibm-web-bnd.xmi

Creating a JDBC Provider for an Oracle Database

SQL Server driver is built-in for WebSphere. however, the Oracle driver must be configured as a JDBC provider.

1. From the left-hand side panel, open **Resources** > **JDBC** > **JDBC Providers**.
2. Click **New**.
3. In Step 1, choose the following:



JDBBProvider.gif

4. In Step 2, specify the location of the `ojdbc14.jar` file. The JAR can be copied from Composer's tomcat/lib directory to a location local to the WebSphere server.

Creating Data Sources

1. On the left-hand side panel, open **Resources** >**JDBC** > **Data sources**.
2. Click **New**.
3. Enter anything you like under **Data source name**.
4. Under **JNDI name**, enter the name that matches the one given in the Connection Profile Editor. Hit **Next**.
5. For the **Select JDBC** provider step, choose **WebSphere embedded ConnectJDBC** driver for MS SQL Server for SQL Server, or **Oracle JDBC** driver for Oracle. Hit **Next**.
6. Enter the database name, host name and port of the database server. Click **Next**. Click **Finish** on the summary page.
7. Next , you must specify the username and password for the database connection. Click on the data source that was just created and then click on the **Custom Properties** link.
8. Create two new properties, called user and password, and specify the credentials for the database.
9. After saving the data source, use the **Test Connection** button to test.

10. Use the **Connection Pool Properties**, link to customize the pooling settings. Refer to the WebSphere documentation for details.

The following items are generated by Composer's WAR export functionality and require no user action.

WEB-INF/web.xml is required, similar to JBoss. <resource-ref id="ResourceRef_1276009394684"> <res-ref-name>jdbc/pooledDS</res-ref-name> <res-type>javax.sql.DataSource</res-type> <res-auth>Container</res-auth> </resource-ref> WEB-INF/ibm-web-bnd.xml does the same thing as jboss-web.xml does for JBoss... <?xml version="1.0" encoding="UTF-8"?> <webappbnd:WebAppBinding xmi:version="2.0" xmlns:xmi= "[http://www.omg.org/XMI" http://www.omg.org/XMI"]; xmlns:webappbnd="webappbnd.xmi" xmi:id="WebAppBinding_1276009185886" virtualHostName="default_host"> <webapp href="WEB-INF/web.xml#WebApp_ID"/> <resRefBindings xmi:id="ResourceRefBinding_1276009394684" jndiName="jdbc/pooledDS"> <bindingResourceRef href="WEB-INF/web.xml#ResourceRef_1276009394684"/> </resRefBindings> </webappbnd:WebAppBinding>