UsingDataSources

Permalink to this page: https://cwiki.apache.org/confluence/x/SzElBg

How do I use DataSources with Tomcat?

When developing JavaEE web applications, the task of database connection management can be daunting. Best practice involves using a JavaEE DataSource to provide connection pooling, but configuring DataSources in web application servers and connecting your application to them is often a cumbersome process and poorly documented.

The usual procedure requires the application developer to set up a DataSource in the web application server, specifying the driver class, JDBC URL (connect string), username, password, and various pooling options. Then, the developer must reference the DataSource in his application's web.xml configuration file, and then access it properly in his servlet or JSP. Particularly during development, setting all of this up is tedious and error-prone.

With Tomcat the process is vastly simplified. Tomcat allows you to configure DataSources for your JavaEE web application in a context.xml file that is stored in your web application project. You don't have to mess with configuring the DataSource separately in the Tomcat server.xml, or referencing it in your application's web.xml file. Here's how:

Install the JDBC Driver

Install the .jar file(s) containing the JDBC driver in Tomcat's `$CATALINA_BASE/lib` folder. You do not need to put them in your application's `WEB-INF/lib` folder. When working with JavaEE DataSources, the web application server manages connections for your application.

Create META-INF/context.xml

In the root of your web app directory structure, create a folder named META-INF (all caps). Inside that folder, create a file named context.xml that contains a Resource like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>

  <Resource name="jdbc/WallyDB" auth="Container"
            type="javax.sql.DataSource" username="wally" password="wally"
            driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
            url="jdbc:sqlserver://localhost;DatabaseName=mytest;SelectMethod=cursor;"
            maxActive="8"
            />

</Context>
```

This example shows how to configure a DataSource for a SQL Server database named mytest located on the development machine. Simply edit the Resource name, driverClassName, username, password, and url to provide values appropriate for your JDBC driver.

Access the DataSource in Your Application

From a Servlet

Here's how you might access the data in a servlet:

```
  InitialContext ic = new InitialContext();
  DataSource ds = (DataSource) ic.lookup("java:comp/env/jdbc/WallyDB");
  Connection c = ds.getConnection();
  ...
  c.close();
```

Notice that, when doing the DataSource lookup, you must prefix the JNDI name of the resource with *java:comp/env/*

Sample Project

Here's a sample web application project that shows where all the files go. This one shows how to access data from from a JSP page: datasourcedemo.war

Known-Working examples for other Databases

```
    <Resource name="jdbc/denali" auth="Container" type="javax.sql.DataSource"
              username="denali" url="jdbc:postgresql://localhost:5432/demo"
              factory="org.apache.commons.dbcp.BasicDataSourceFactory"
              driverClassName="org.postgresql.Driver"
              maxActive="20" maxIdle="10"/>
```

```
    <Resource name="jdbc/ccsdatasource" auth="Container" type="javax.sql.DataSource"
              username="ccs" password="secret" url="jdbc:mysql://localhost:3306/ccs"
              driverClassName="com.mysql.jdbc.Driver"
              maxActive="20" maxIdle="10"/>
```

Please Note

This technique is Tomcat-specific. If you deploy your web application to another application server, you will need to configure the database according to your application server's documentation, and reference it in your application's web.xml.