Configuring JDBC in TomEE

You can configure a JDBC driver for the entire TomEE application server or for a specific webapp. Regardless of which way you choose to do it, the configuration syntax is the same.

NOTE: You can also configure a JDBC `DataSource` directly within an `@Resource` annotation but this is not recommended and is not covered here.

The TomEE Resource Configuration Element

Here is a partial example of how you would configure a JDBC data source for Oracle.

```
<Resource id="Oracle Database" type="DataSource">
    // driver properties go here
</Resource>
```

We'll fill in the driver properties in a bit, for now let's focus on the `<Resource>` element's attributes `id` and `type`.

NOTE: Within the `<Resource>` element values for `id` and `type` attributes allow for spaces and are not case sensitive.

JDBC Resource Attributes

There are several possible attributes that can be set on the `<Resource>` element, but the most important are the `id` and the `type`. The `id` is the name you will use in the `@Resoruce` annotation in your source code while the `type` identifies the Java type being injected. The type "`DataSource`" is shorthand for the `javax.sql.DataSource` type.
For example, the JdbcServet example shown below uses these attribute values as follows:

```
package org.superbiz;
import javax.servlet.http.HttpServlet;
import javax.servlet.*;
import java.io.*;

import javax.annotation.Resource;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.SQLException;

public class JdbcServlet extends HttpServlet {
        private static final long serialVersionUID = 8229511817604046163L;

        @Resource(name = "Derby Database")
        private DataSource dataSource;
```

```
        protected void doGet(HttpServletRequest request,
HttpServletResponse response)
                        throws ServletException, IOException {
                PrintWriter writer = response.getWriter();


                try(final Connection connection =
dataSource.getConnection()){

                        // do something

                } catch (SQLException e) {

                }

                writer.print("\n Unable to Connect");

        }
}
```

The `@Resource` annotation specifies the resource `id`, in this case "Oracle Database", which matches the `id` of the `<Resource>` element shown earlier. The field type is `javax.sql.DataSource` which matches the type attribute from the `<Resource>` element as well.

JDBC Resource Properties

In addition to the `<Resource>` element attributes you must also specify a number of properties which are listed within the body of element. Each property is shown on its own line. The name of the attribute is separated from the value using either a space as shown below, or an "=" sign.

```
<Resource id="Oracle Database" type="DataSource">
    JdbcDriver  oracle.jdbc.OracleDriver
    JdbcUrl jdbc:oracle:thin:@localhost:1521:orcl
    UserName    scott
    Password    tiger
</Resource>
```

JDBC drivers almost always require you to set at least four properties: `JdbcDriver`, `JdbcUrl`, and driver-specific attributes. Here is what these properties mean:

JdbcDriver

The actual JDBC driver class for a specific database. For example, Oracle uses the `oracle.jdbc.OracleDriver` while PostgreSQL uses the `org.postgresql.Driver` type.

JdbcUrl

The database URL includes the `jdbc` schema, the database brand and model, the path to the database (either in a local directory or JAR file), and the database name (whatever you named

your database). For example, the `JdbcUrl` for an Oracle driver might look like this "`jdbc:oracle:thin:@localhost:1521:orcl`" while a MySQL JdbcUrl might look like this: "`jdbc:mysql://localhost/test`".

Database specific attributes

In addition to the properties listed above, TomEE supports a plethora of other properties common to databases. These range from setting the JDBC login credentials (`userName/password`), connection pool size, to transaction management, to isolation levels.

Specific JDBC Configurations for TomEE

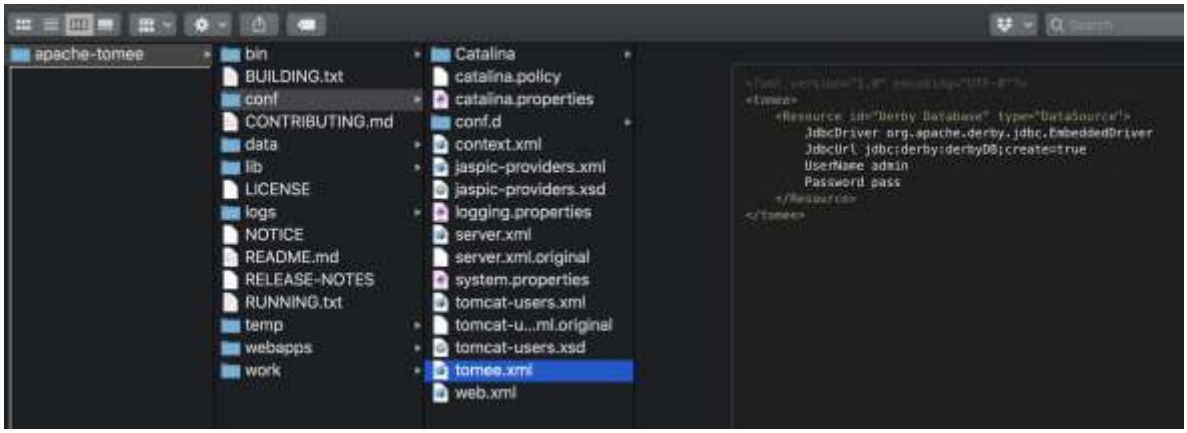
Every JDBC driver is configured differently – you can consult your JDBC driver documentation to figure out what properties you need to specify. There is a nice document, "Common Database Configurations", on the TomEE web site that shows the `<Resource>` configuration using several different drivers.

JDBC Configuration for the TomEE Server

If you wish to configure a JDBC driver for the entire TomEE server than you can add the `<Resource>` element directly under the `<tomee>` element of the `tomee.xml` file. Below is an example of a JDBC configuration for the Derby Embedded database.
`<tomee>`

```
    <Resource id="Derby Database" type="DataSource">
       JdbcDriver org.apache.derby.jdbc.EmbeddedDriver
       JdbcUrl jdbc:derby:derbyDB;create=true
       UserName admin
       Password pass
    </Resource>
</tomee>
```

The `tomee.xml` configuration file is located at `<apache-tomee>/conf/tomee.xml` as illustrated below.

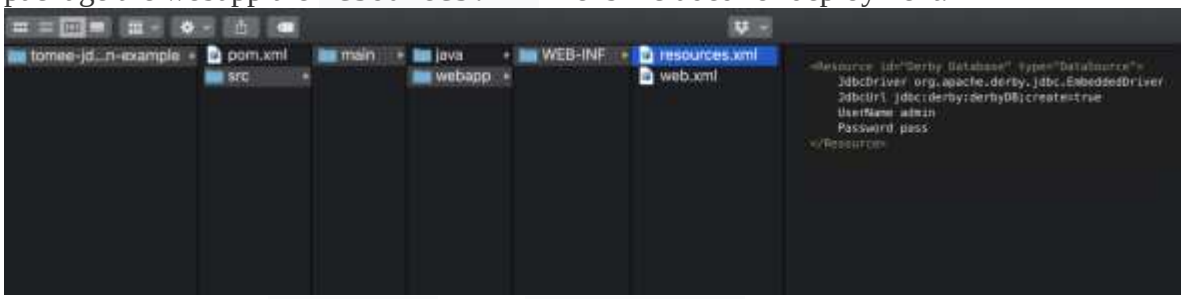


JDBC Configuration for a Webapp in TomEE

If you wish to configure a JDBC driver for a specific webapp than you add the `<Resource>` element as the root of the `resources.xml` file like this.

```
<Resource id="Derby Database" type="DataSource">
    JdbcDriver org.apache.derby.jdbc.EmbeddedDriver
    JdbcUrl jdbc:derby:derbyDB;create=true
    UserName admin
    Password pass
</Resource>
```

The `resources.xml` file should be created and stored under the `webapp/WEB-INF/resources.xml` in the source file for you webapp as shown below so that when you package the webapp the `resources.xml` file is included for deployment.



The declaration of a `<Resource>` in the `resources.xml` file will override any setting for the same `<Resource>` `id` in the `tomee.xml` file. In addition, you can set the TomEE system properties for a specific JDBC connection and that will take precedence over both the `tomee.xml` and the `resources.xml` configurations.

The JDBC Configuration Example Code for TomEE

You can find the source code on GitHub at the tomitribe/tomee-jdbc-configuration-example project.

NOTE: The example uses Java 8 when executing the Maven `pom.xml` file. Other versions of Java may not work.

Simply clone the project from GitHub using the following command:

```
git clone https://github.com/tomitribe/tomee-jdbc-configuration-example.git
```

Next you can download, install, and run TomEE with a single Maven command by navigating into the project directory and executing the following command:

```
cd tomee-jdbc-configuration-example
```

```
mvn -e clean install tomee:run
```

NOTE: The `-e` option tells Maven to output any execution errors when it starts and runs the TomEE server.

The `pom.xml` file includes the TomEE Maven Plugin which takes care of downloading TomEE, installing the server, deploying the webapp and starting the Server.

NOTE: You can stop the TomEE server by typing `exit` or `quit` at the prompt.

To test that your web app was deployed and that JDBC was configured correctly, use your web browser or `cUrl` (on a new terminal) to send a `GET` request to the following address:

```
curl http://localhost:8080/jdbc-servlet/




 JDBC Connection is valid @ timestamp = 1573829415802
```

If the output says that the connection "is valid" than you have successfully configured JDBC and connected to the database. Everytime you send this `GET` request the timestamp is updated. The example injects a Derby DataSource into a raw Servlet using the `@Resources` annotation. The Servlet uses the injected Derby DataSource to create a `Connection` object and test if the connection is valid as shown in the full code listing below.

```
package org.superbiz;
import javax.servlet.http.HttpServlet;
import javax.servlet.*;
import java.io.*;

import javax.annotation.Resource;
import javax.sql.DataSource;
import java.sql.Connection;
import java.sql.SQLException;

public class JdbcServlet extends HttpServlet {
        private static final long serialVersionUID = 8229511817604046163L;

        @Resource(name = "Derby Database")
        private DataSource dataSource;

        protected void doGet(HttpServletRequest request,
HttpServletResponse response)
```

```
                        throws ServletException, IOException {
                PrintWriter writer = response.getWriter();


                try(final Connection connection =
dataSource.getConnection()){

                        boolean valid = connection.isValid(1000);
                        writer.print("\n JDBC Connection is " +
(valid==true?"valid":"NOT valid") + " @ timestamp = "+
System.currentTimeMillis());

                        return;

                } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

                writer.print("\n Unable to Connect");

        }
}
```

For this example the Derby JDBC Driver was configured using the `webapp/WEB-INF/resources.xml` configuration file.

As you can see, configuring a JDBC driver for use in TomEE really is not that complicated. All you need to know are the properties the driver requires and then to decide if you want the JDBC driver configured across the entire server or only for a specific webapp.