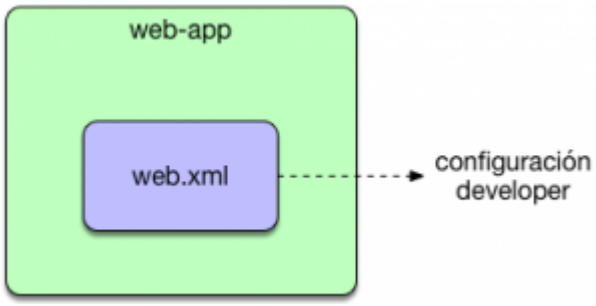
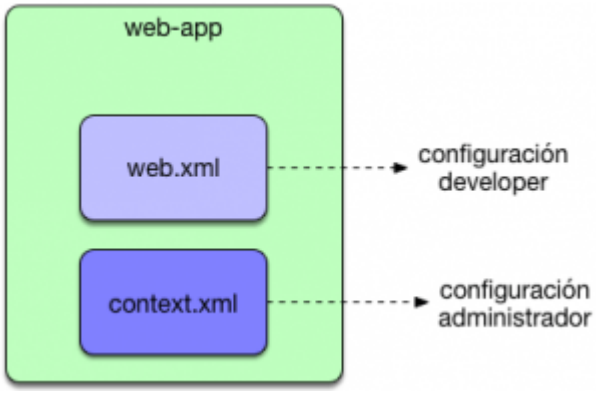
El uso de Tomcat context xml como fichero para configurar funcionalidad de servidor es muy habitual. Normalmente la configuración de una aplicación web se realiza a través el web.xml .



Pero hay algunas partes de la configuración que están más ligadas a las tareas de administración y no usan el web.xml ya que este es neutro y no pertenece a ningún tipo de servidor en concreto. Así pues cada fabricante define el suyo. Vamos a ver como usar el fichero context.xml en Tomcat para dar de alta una fuente de datos o Datasource.



Utilizando Tomcat context xml

Es momento de crear este fichero en nuestra aplicación web y abordar su configuración. Para ello añadimos el fichero en la carpeta META-INF de la aplicación web:

Acabamos de añadir contenido al fichero y configurar una fuente de datos o Datasource:

```
&amp;amp;lt;?xml version="1.0" encoding="UTF-8"?&amp;amp;gt;

&amp;amp;lt;Context&amp;amp;gt;
    &amp;amp;lt;Resource name="jdbc/cursosDS"
type="javax.sql.DataSource" maxActive="100" maxIdle="30"
maxWait="10000" url="jdbc:mysql://localhost:3306/java"
driverClassName="com.mysql.jdbc.Driver" username="root"
password="mysql" /&amp;amp;gt;

&amp;amp;lt;/Context&amp;amp;gt;
```

Acabamos de configurar el DataSource con los siguientes parámetros:

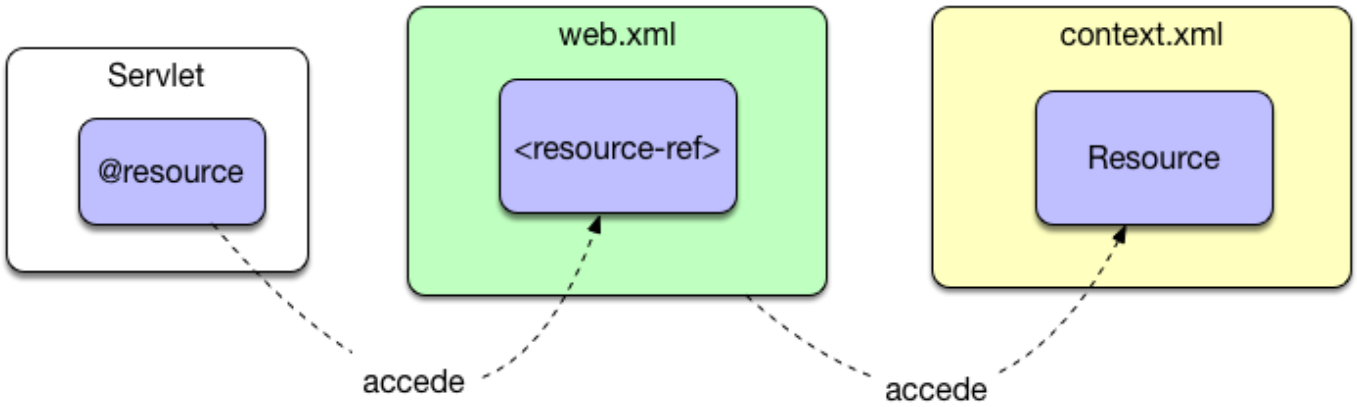
1. name: Nombre del Datasource o fuente de datos
2. type: type de Datasource en este caso SQL
3. url: Dirección del servidor y la base de datos de Mysql
4. driverClassName: Tipo de driver
5. username: Usuario de conexión

6. password: Clave de conexión

   Nos queda modificar el web.xml para que enlace con el DataSource construido:

```
 &amp;amp;lt;resource-ref&amp;amp;gt;

    &amp;amp;lt;res-ref-name&amp;amp;gt;jdbc/cursosDS&amp;amp;lt;/res-ref-name&amp;amp;gt;
    &amp;amp;lt;res-type&amp;amp;gt;java.sql.DataSource&amp;amp;lt;/res-type&amp;amp;gt;
    &amp;amp;lt;res-auth&amp;amp;gt;Container&amp;amp;lt;/res-auth&amp;amp;gt;
  &amp;amp;lt;/resource-ref&amp;amp;gt;
```

   Una vez hecho esto el siguiente paso es construir un Servlet en nuestra aplicación que sea capaz de obtener una referencia al DataSource.



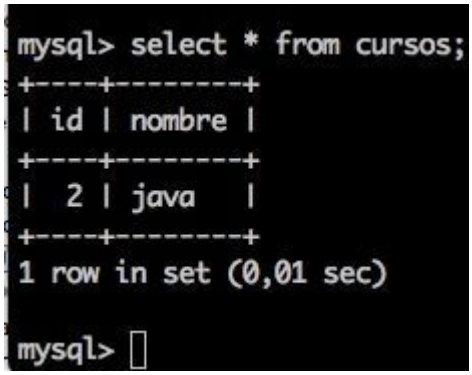
   Este Servlet lanzará una consulta de inserción:

```
package com.arquitecturajava.ejemplo1;
```

```
import java.io.IOException;
import java.sql.Connection;
import java.sql.Statement;

import javax.annotation.Resource;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.sql.DataSource;

@WebServlet("/ServletInsercion")
public class ServletInsercion extends HttpServlet {
        private static final long serialVersionUID = 1L;

        @Resource(name = "jdbc/cursosDS")
        private DataSource fuente;

        protected void doGet(HttpServletRequest request,
HttpServletResponse response)
                        throws ServletException, IOException {

                String sql = "insert into cursos  values (2,'java') ";

                try (Connection conn = fuente.getConnection();
Statement stmt = conn.createStatement();) {

                        stmt.executeUpdate(sql);

                } catch (Exception se) {
```

```
                    throw new RuntimeException("error SQL", se);
                }
        }

}
```

Hemos construido un Servlet que hace uso de una anotación @Resource para referirse al DataSource que acabamos de configurar entre el web.xml y el context.xml. Nos queda invocar el Servlet y que nos inserte la información en la base de datos.



Recordemos que para que la aplicación funcione debemos agregar el driver JDBC de mysql. Acabamos de configurar el fichero Tomcat context xml con un Datasource.

Otros artículos relacionados:

1. JDBC Batch y rendimiento
2. JDBC ResultSet Types y su funcionamiento
3. Java LinQ con JinQ y Java Persistence API
4. JPA Composite Key y business objects
5. Referencia de Tomcat