Tabla de Contenidos

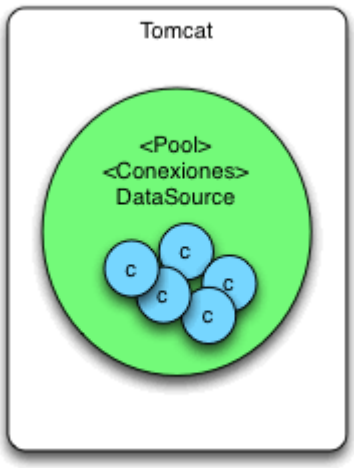
- Configuración Java DataSource
- Recursos y web.xml
- @Resource y Servlets

El concepto de Java DataSource es común en la mayor parte de las aplicaciones Java EE y hace referencia a un conjunto de conexiones a base de datos (pool de conexiones) creadas por el servidor que se alquilan durante un determinado tiempo a cada Servlet JSP o componente que lo requiera.

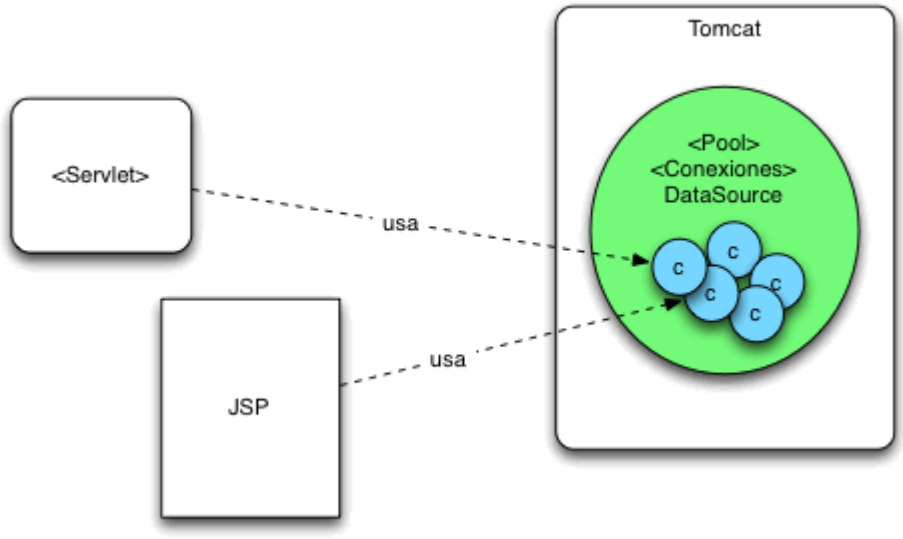
CURSO SPRING BOOT

GRATIS

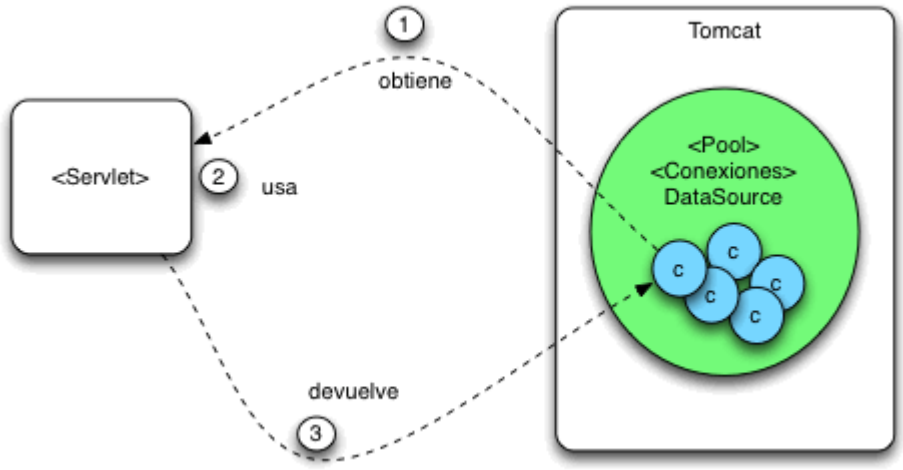
APUNTATE!!



Estas conexiones son utilizadas por las distintos elementos de la aplicación.



El uso de estas conexiones es durante un tiempo determinado en el cual el Servlet JSP o componente la necesitan. Una vez que han terminado de usarla la devuelven al pool y así permiten su reutilización.



Configuración Java DataSource

Vamos a configurar el pool de conexiones utilizando un Tomcat .Para ello deberemos abrir el fichero server.xml de nuestro servidor. Este fichero contiene una sección de contexto por cada aplicación web desplegada. En nuestro caso la aplicación web se llama “Recursos” y por lo tanto tiene un contexto como el siguiente:

```
&amp;lt;Context docBase="Recursos" path="/Recursos" reloadable="true"
source="org.eclipse.jst.jee.server:Recursos"&amp;gt;

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

Dentro de la etiqueta de contexto añadimos un “Recurso ” (Resource) que nos conecta a la base de datos.

```
&amp;lt;Context docBase="Recursos" path="/Recursos" reloadable="true"
source="org.eclipse.jst.jee.server:Recursos"&amp;gt;
&amp;lt;Resource name="jdbc/miDataSource" auth="Container"
type="javax.sql.DataSource"
maxActive="100" maxIdle="30" maxWait="10000"
username="root" password="" driverClassName="com.mysql.jdbc.Driver"
url="jdbc:mysql://localhost:3306/miBaseDeDatos"/&amp;gt;
&amp;lt;/Context&amp;gt;
```

En este caso hemos usado un driver de MySQL que deberemos configurar en el propio Tomcat en la carpeta lib del servidor.

Recursos y web.xml

Una vez dado de alta el pool de conexiones a nivel de server.xml debemos modificar el fichero web.xml y registrarlo como un recurso accesible.

```
&amp;lt;?xml version="1.0" encoding="UTF-8"?&amp;gt;
&amp;lt;web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
version="3.0"&amp;gt;
&amp;lt;display-name&amp;gt;Recursos&amp;lt;/display-name&amp;gt;
&amp;lt;resource-ref&amp;gt;
&amp;lt;description&amp;gt;fuente datos
&amp;lt;/description&amp;gt;
&amp;lt;res-ref-name&amp;gt;jdbc/miDataSource&amp;lt;/res-ref-
name&amp;gt;
&amp;lt;res-type&amp;gt;java.sql.DataSource&amp;lt;/res-type&amp;gt;
&amp;lt;res-auth&amp;gt;Container&amp;lt;/res-auth&amp;gt;

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

&amp;lt;/web-app&amp;gt;
```

@Resource y Servlets

Hecho esto podemos usar directamente el recurso usando la anotación de @Resource vamos a verlo a través de un simple Servlet.

```
&amp;lt;pre&amp;gt;
package com.arquitecturajava;
```

```
import java.io.IOException;
import java.sql.Connection;
import java.sql.SQLException;
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("/WebJDBC003")
public class WebJDBC003DataSourceRecurso extends HttpServlet {
private static final long serialVersionUID = 1L;

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

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

Connection conn = null;
Statement stmt = null;
try{

System.out.println("Conectando a la base de datos");
conn = fuente.getConnection();

stmt = conn.createStatement();
```

```
String sql = "INSERT INTO Personas " +
"VALUES ('pepe3')";
stmt.executeUpdate(sql);
}catch(SQLException se){

se.printStackTrace();
}catch(Exception e){

e.printStackTrace();
}finally{

try{
if(stmt!=null)
conn.close();
}catch(SQLException se){
}
try{
if(conn!=null)
conn.close();
}catch(SQLException se){
se.printStackTrace();
}
}
}
}
```

Como vemos ahora es muy sencillo inyectar un DataSource a un Servlet o ha cualquier otro componente como un EJB simplemente nos apoyamos en la anotación @Resource. Recordemos que si usamos JPA podemos usar las anotaciones de @PersistenceContext y @PersistenceUnit.

CURSO SPRING BOOT

GRATIS

APUNTATE!!

Otros artículos relacionados:

Introducción a JPA

Libro JPA

JPA OneToMany