
Práctica 7.8: *Pool de conexiones con JNDI y Servlets 3.0*

En las aplicaciones con acceso a datos, crear una nueva conexión para cada petición es algo totalmente ineficiente debido a que el coste que conlleva crear la conexión e inicializarla suele ser mayor que el de la operación que queremos realizar.

La opción de mantener una conexión abierta compartida puede acarrear problemas de concurrencia.

Una solución habitual a estos problemas consiste en mantener un grupo, *pool*, de conexiones abiertas que funciona de la siguiente manera: Cuando necesitamos una conexión la solicitamos al *pool*, este busca una que esté libre y nos la da, apuntando que la tenemos nosotros y que deja de estar libre. Tras realizar la operación cerramos la conexión. El *pool* recibe esta petición de cierre pero no cierra la conexión, sino que la deja abierta y la vuelve a marcar como libre para la siguiente petición. Esta reutilización de conexiones es más eficiente que andar abriendo nuevas conexiones para cada petición.

Tomcat 7.0 permite utilizar un *pool* de conexiones utilizando *JNDI* (*Java Directory and Naming Interface*). En el siguiente apartado vamos a desarrollar un ejemplo para comprobar cómo funcionaría.

1. Pool de conexiones con JNDI.

- 1.1. Crea un nuevo *Dynamic Web Project*, en *Eclipse* llamado **PoolConexiones**. Esta aplicación pedirá un usuario y contraseña y comprobará si existen en la base de datos, en tal caso creará una variable de sesión para almacenar el usuario.
- 1.2. Ejecuta el siguiente *script* en la base de datos **TiendaLibros**, con él crearemos una tabla de usuarios y contraseñas.

```
use TiendaLibros;

drop table if exists usuarios;

create table usuarios(
  usuario char(16) not null,
  password char(41) not null,
  primary key (usuario));

insert into usuarios values ('usuario1', password('usuario1')),
  ('administrador1', password('administrador1'));

select * from usuarios;
```

- 1.3. El siguiente paso consistirá en definir el recurso *JNDI DataSource* en *Tomcat* en el fichero de contexto **context.xml** de la instancia del servidor de **tomcat** definida en **eclipse**. Este fichero permite definir configuraciones válidas para todos los contextos de *Tomcat*, ver Figura 1:



Figura 1: *Pool* de conexiones

```
<Resource name="jdbc/mysql_tiadalibros" auth="Container" type="javax.sql.DataSource"
    maxActive="100" maxIdle="30" maxWait="10000" removeAbandoned="true"
    username="root" password="despliegue" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost/TiendaLibros" />
```

1.4. A la hora de definir el recurso se pueden establecer diferentes parámetros:

- *Resource name*: Establece el nombre del recurso disponible para las aplicaciones.
- *type*: Establece el tipo del recurso disponible para las aplicaciones.
- *url*: La cadena de conexión de la base de datos.
- *driverClassName*: El nombre de la clase del *driver JDBC*.
- *username*: El nombre de usuario para acceder a la base de datos.
- *password*: La contraseña del usuario para acceder a la base de datos.
- *maxActive*: El número máximo de conexiones en el pool de conexiones.
- *maxIdle*: El número máximo de conexiones inactivas a retener en el *pool* de conexiones.
- *maxWait*: El tiempo máximo a esperar para obtener una conexión, en mili segundos.
- *removeAbandoned*: Si *removeAbandoned = true* entonces cuando haya pocas conexiones disponibles en el *pool* de conexiones se recuperará y reciclará cualquier conexión abandonada que se encuentre.
- etc.

1.5. A continuación añade una página **index.html**, a la carpeta **WebContent**, que será el punto de entrada a la aplicación:

```
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
</head>
<body>
    <h2>Login</h2>
    <form method="get" action="login">
    <table>
        <tr>
            <td>Introduce tu usuario:</td>
            <td><input type='text' name='usuario' /></td>
        </tr>
        <tr>
            <td>Introduce tu contraseña:</td>
```

```
        <td><input type='password' name='password' /></td>
    </tr>
</table>
<br />
<input type="submit" value='LOGIN' />
<input type="reset" value='LIMPIAR' />
</form>
</body>
</html>
```

1.6. Ahora añade el *servlet* **LoginServlet.java** con el siguiente código:

```
import java.io.*;
import java.util.*;
import java.util.logging.*;
import javax.naming.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.sql.*;
import javax.sql.*;

public class LoginServlet extends HttpServlet {

    private DataSource pool; // Pool de conexiones a la base de datos

    @Override
    public void init(ServletConfig config) throws ServletException {
        try {
            // Crea un contexto para poder luego buscar el recurso DataSource
            InitialContext ctx = new InitialContext();
            // Busca el recurso DataSource en el contexto
            pool = (DataSource)ctx.lookup("java:comp/env/jdbc/mysql_tiadalibros");
            if (pool == null)
                throw new ServletException("DataSource desconocida 'mysql_tiadalibros'");
        } catch (NamingException ex) {
            Logger.getLogger(LoginServlet.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();

        Connection conn = null;
        Statement stmt = null;
        try {
            out.println("<html><head><title>Login</title></head><body>");
        }
```

```
out.println("<h2>Login</h2>");

conn = pool.getConnection(); // Obtiene un conexión del pool
stmt = conn.createStatement();

// recupera los parámetros de la petición request usuario y password
String usuario = request.getParameter("usuario");
String password = request.getParameter("password");
boolean noUsuario = usuario != null && ((usuario=usuario.trim()).length()>0);
boolean noPwd = password != null && ((password=password.trim()).length()>0);

// Valida los parámetros de la petición request
if (!noUsuario) {
    out.println("<h3>Debes introducir tu usuario</h3>");
} else if (!noPwd) {
    out.println("<h3>Debes introducir tu password</h3>");
} else {
    // Verifica que existe algún usuario con ese login y password
    StringBuilder sqlStr = new StringBuilder();
    sqlStr.append("SELECT * FROM usuarios WHERE ");
    sqlStr.append("STRCMP(usuarios.usuario, '")
        .append(usuario).append("' ) = 0 ");
    sqlStr.append("AND STRCMP(usuarios.password, PASSWORD('")
        .append(password).append("' ) = 0 ");

    ResultSet rset = stmt.executeQuery(sqlStr.toString());

    if (!rset.next()) { // si no está vacío el resulset
        out.println("<h3>Nombre o contraseña incorrecta</h3>");
        out.println("<p><a href='index.html'>Vuelve a la página login</a></p>");
    } else {

        // Crea una nueva sesión y guarda el usuario como variable de sesión
        // Primero, invalida la sesión si existe
        HttpSession session = request.getSession(false);
        if (session != null) {
            session.invalidate();
        }
        session = request.getSession(true);
        synchronized (session) {
            session.setAttribute("usuario", usuario);
        }

        out.println("<p>Hola, " + usuario + "!</p>");
        out.println("<p><a href='hazalgo'>Haz algo</a></p>");
    }
}

out.println("</body></html>");

} catch (SQLException ex) {
```

```
        out.println("<h3>Servicio no disponible</h3></body></html>");
        Logger.getLogger(LoginServlet.class.getName()).log(Level.SEVERE, null, ex);
    } finally {
        out.close();
        try {
            if (stmt != null) stmt.close();
            if (conn != null) conn.close(); // Devuelve la conexión al pool
        } catch (SQLException ex) {
            Logger.getLogger(LoginServlet.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    doGet(request, response);
}
}
```

- 1.7. Observa en el código como se solicita una conexión al *pool* de conexiones y como se devuelve al pool cuando se cierra.
- 1.8. Fijate como se crea una instancia de *InitialContext* que permitirá buscar el recurso *DataSource* `jdbc/mysql_tiadalibros`. Los recursos definidos están ubicados en la sección `java:comp/env` del directorio de nombres *JNDI*.

```
InitialContext ctx = new InitialContext();
pool = (DataSource)ctx.lookup("java:comp/env/jdbc/mysql_tiadalibros");
```

- 1.9. Crea el *servlet* **HazAlgo.java**, con el siguiente código:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class HazAlgo extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();

        try {
            out.println("<html><head><title>Haz algo</title></head><body>");
            out.println("<h2>Haz algo...</h2>");

            // Recupera el nombre de usuario
            String usuario;
```

```
        HttpSession session = request.getSession(false);
        if (session == null) {
            out.println("<h3>No has iniciado sesión</h3>");
        } else {
            synchronized (session) {
                usuario = (String) session.getAttribute("usuario");
            }

            out.println("<table>");
            out.println("<tr>");
            out.println("<td>Usuario:</td>");
            out.println("<td>" + usuario + "</td></tr>");
            out.println("<tr>");
            out.println("</table>");

            out.println("<p><a href='logout'>Logout</a></p>");
        }
        out.println("</body></html>");
    } finally {
        out.close();
    }
}
```

1.10. A continuación añade el *servlet* **LogoutServlet.java** con el siguiente código:

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class LogoutServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();

        try {
            out.println("<html><head><title>Logout</title></head><body>");
            out.println("<h2>Logout</h2>");
            HttpSession session = request.getSession(false);
            if (session == null) {
                out.println("<h3>No has iniciado sesión</h3>");
            } else {
                session.invalidate();
                out.println("<p>Adios</p>");
                out.println("<p><a href='index.html'>Login</a></p>");
            }
            out.println("</body></html>");
        } finally {
            out.close();
        }
    }
}
```

```
    }  
  }  
}
```

- 1.11. Finalmente añade la siguiente información al descriptor de despliegue de la aplicación, **web.xml**:

```
<servlet>  
  <servlet-name>LoginServlet</servlet-name>  
  <servlet-class>LoginServlet</servlet-class>  
</servlet>  
<servlet>  
  <servlet-name>HazAlgoServlet</servlet-name>  
  <servlet-class>HazAlgo</servlet-class>  
</servlet>  
<servlet>  
  <servlet-name>LogoutServlet</servlet-name>  
  <servlet-class>LogoutServlet</servlet-class>  
</servlet>  
<servlet-mapping>  
  <servlet-name>LoginServlet</servlet-name>  
  <url-pattern>/login</url-pattern>  
</servlet-mapping>  
<servlet-mapping>  
  <servlet-name>HazAlgoServlet</servlet-name>  
  <url-pattern>/hazalgo</url-pattern>  
</servlet-mapping>  
<servlet-mapping>  
  <servlet-name>LogoutServlet</servlet-name>  
  <url-pattern>/logout</url-pattern>  
</servlet-mapping>  
<session-config>  
  <session-timeout>30</session-timeout>  
</session-config>  
<welcome-file-list>  
  <welcome-file>index.html</welcome-file>  
</welcome-file-list>
```

- 1.12. Comprueba el funcionamiento de la misma, como valida usuario y contraseña contra la base de datos utilizando una conexión del pool, y cómo utilizan la variable de sesión los *servlets*, ver Figura 2:



Figura 2: Funcionamiento aplicación

2. Servlets 3.0.

La especificación *Servlet 3.0* introduce la posibilidad de facilitar el despliegue de los *servlets* introduciendo anotaciones dentro del código del *servlet* en vez de en el descriptor de despliegue.

- `@WebServlet`: Para definir un *servlet*.
 - `@WebInitParam`: Para definir los parámetros de inicialización de un *servlet*.
 - `@WebListener`: Para definir un *listener*.
 - `@WebFilter`: Para definir un filtro.
 - `@MultipartConfig`: Para subida múltiple de archivos.
- 2.1. Modifica los tres *servlets* de la aplicación y para que incluyan las anotaciones relativas al despliegue. A continuación se muestran las modificaciones necesarias en el *servlet* **LoginServlet.java**:
- ```
...
import javax.servlet.annotation.WebServlet;

@WebServlet(
 name = "LoginServlet",
 urlPatterns = {"/login"})
public class LoginServlet extends HttpServlet {
 ...
```
- 2.2. Una vez realizadas las anotaciones en los tres *servlets* podemos eliminar la información relativa a los *servlets* que añadimos al descriptor de despliegue en su momento.
- 2.3. Para finalizar comprueba cómo la aplicación sigue funcionando de igual forma.

◇