# Ejercicios sesión 2: Conceptos básicos de JPA

## Índice

# 1. Servlet DoAction

Vamos a modificar la aplicación web para que sea más sencillo añadir nuevas funcionalidades. En la página HTML vamos a incluir un nuevo campo de texto en el que el usuario escriba la acción que quiere realizar. También vamos que cambiar el servlet para que lea la acción y llame al método (privado) correspondiente que procesa esa acción.

1. Copia el proyecto `jpa-sesion1-web` y pégalo con el nombre `jpa-sesion2-web`. Aunque esto debería ser suficiente, Eclipse tiene un bug y no configura correctamente el nuevo proyecto. Lo puedes comprobar escogiendo la opción **Add and Remove Projects...** del servidor y comprobando que en la caja de la izquierda aparece el proyecto recién creado con el mismo nombre que tenía anteriormente.

2. Para arreglarlo, debes editar el fichero `.settings/org.eclipse.wst.common.component` y sustituir en él el nombre `jpa-sesion1-web` por `jpa-sesion2-web`. Debe quedar como sigue:

```
<?xml version="1.0" encoding="UTF-8"?>
<project-modules id="moduleCoreId" project-version="1.5.0">
   <wb-module deploy-name="jpa-sesion2-web">
      <wb-resource deploy-path="/" source-path="/WebContent"/>
      <wb-resource deploy-path="/WEB-INF/classes"
source-path="/src"/>
      <wb-resource deploy-path="/WEB-INF/classes"
source-path="/etc"/>
      <property name="java-output-path" value="build/classes"/>
      <property name="context-root" value="jpa-sesion2-web"/>
   </wb-module>
</project-modules>
```

3. Prueba de nuevo a añadir el nuevo proyecto al servidor Tomcat. Ahora ya debe aparecer con el nombre correcto. Añade el proyecto y comprueba que la aplicación web (ahora en **http://localhost:8080/jpa-sesion2-web/**) funciona correctamente.

4. Cambia el fichero `index.html`, modificando el nombre del servlet y añadiendo el nuevo parámetro `accion` ligado a un nuevo campo de entrada. En ese nuevo campo escribiremos la acción que queremos probar. Así nos será más sencillo añadir nuevas funcionalidades sin tener que retocar demasiado las páginas HTML de la aplicación.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
   <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8">
   <title>Aplicación web Mensajes</title>
</head>
<body>
```

```
   <form action="servlet/DoAction">
   Mensaje:  <input type="text" name="mensaje"> <br>
   Autor: <input type="text" name="autor"><br>
   Acción: <input type="text" name="accion"><br>
   <input type="submit" value="Enviar">
   </form>
</body>
</html>
```

5. Renombra el servlet `AddMensaje` por `DoAction` y reescríbelo como sigue:

```
package es.ua.jtech.jpa;

import java.io.*;
import java.util.Collection;

import javax.persistence.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class DoAction extends HttpServlet {
   private static final long serialVersionUID = 1L;

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

      String accion = request.getParameter("accion");

      if (accion.equals("add")) {
         doAdd(request, response);
      } else if (accion.equals("lista")) {
         doLista(request, response);
      } else if (accion.equals("cuenta")) {
         doCuenta(request, response);
      } else {
         PrintWriter out = response.getWriter();
         out.println("Error. La acción " + accion + " no está
implementada");
      }
   }

   private void doCuenta(HttpServletRequest request,
         HttpServletResponse response) throws ServletException,
IOException {

           // Cuenta los mensajes de un autor

   }

   private void doLista(HttpServletRequest request,
                        HttpServletResponse response)
         throws ServletException, IOException {

      // Lista los mensajes de un autor

   }
```

```
   private void doAdd(HttpServletRequest request,
                      HttpServletResponse response)
         throws ServletException, IOException {

      // Añade un mensaje a un autor. Mismo código que el servlet
AddMensaje
   }
}
```

6. Cambia por último el fichero `web.xml` para reflejar el nuevo nombre del servlet.

7. Comprueba que todo funciona correctamente con la nueva estructura del servlet. Ahora podremos añadir fácilmente nuevas funciones. Sólo tendremos que añadir un nuevo caso en el `switch`, el código de la nueva función en un método privado y una nueva página JSP que haga la presentación de los resultados.

## 2. Construcción de clases DAO

Construye las clases `AutorDAO` y `MensajeDAO`, siguiendo el esquema visto en la clase de teoría. Modifica el método `doAdd()` del servlet para que haga las llamadas a estas clases, en lugar de a las entidades.

## 3. Implementación de funciones adicionales

Implementa las acciones que están vacías en el servlet (`doLista` y `doCuenta`). Implementa las nuevas páginas JSP:

- `addMensaje.jsp`: muestra el nombre del usuario y el texto del mensaje añadido
- `cuentaMensajes.jsp`: muestra el nombre del usuario y el número de mensajes que ha añadido
- `listaMensajes.jsp`: muestra el nombre del usuario y la lista de mensajes que ha añadido (**ya está implementada**)

Y, por último, haz que el resultado de cada acción del servlet muestre la página JSP correspondiente. El método `doAdd()` debería redirigir la petición a la página `addMensaje.jsp` y lo mismo con el resto de métodos.