migsalazar / LDOO

Branch: master ▾ LDOO / lab04 / Create new file Find file History

migsalazar committed on GitHub lab04 Latest commit 75b41c3 3 days ago

..

README.md lab04 3 days ago

README.md

Laboratorio 4

Crear un login simple con MVC

Objetivo: Al finalizar la actividad deberás ser capaz de construir un sistema de autenticación simple basado en el patrón de diseño `MVC` (Model-View-Controler).

Preparación

La descripción de este laboratorio parte del supuesto que ya se cuenta con la versión de NetBeans (EE) y que incluye un contenedor Web GlassFish o Apache Tomcat atendiendo un puerto de escucha 8080.

Actividades

En este laboratorio se realizará lo siguiente:

Crear una mini-aplicación web que funcione con el patrón MVC:

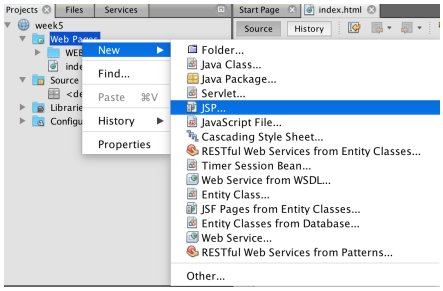
- Crear tres páginas JSP para que funcionen como `View`.
- Crear un servlet que funcionará como `Controller`.
- Crear dos clases de Java que funcionarán como `Model`.

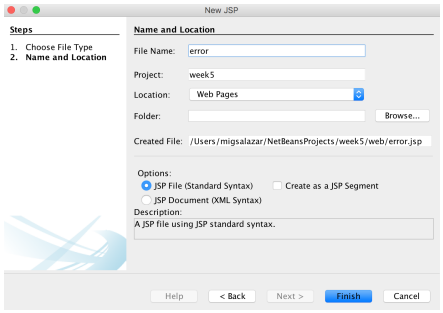
Actividad 1 - Construcción de vistas

1.- Crear un proyecto de tipo `Java Web` en NetBeans.

2.- Elimina el archivo `index.html` que crea por defecto NetBeans y que se encuentra dentro de la carpeta `Web Pages`.

3.- Agrega tres archivos de tipo JSP: `login.jsp`, `success.jsp` y `error.jsp`.

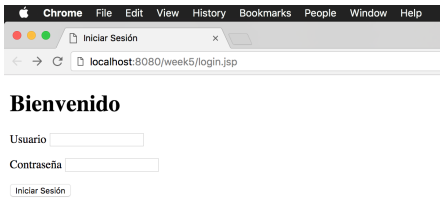




4.- La página `login.jsp` deberá contener lo siguiente:

- Un título `h1` con un mensaje de bienvenida.
- Un formulario.
- El formulario deberá ejecutar una accion `LoginController` el cuál será un servlet que se desarrollará en la siguiente actividad.
- El formulario deberá enviar los datos mediante el método de transferencia `POST` .
- El formulario deberá contener tres `input` : uno de tipo `text` , el segundo de tipo `password` y el tercero de tipo `submit` con el valor `Iniciar sesión` .

La página de `login.jsp` deberá ser tener un aspecto muy similar al de la siguiente imagen:

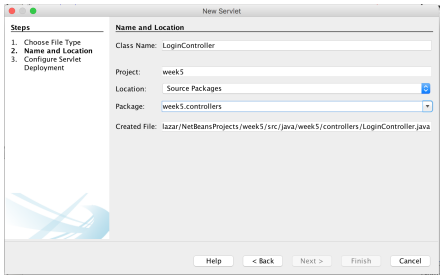


5.- La página `error.jsp` deberá contener un título `h1` con el contenido `Usuario o contraseña incorrectos` . Además un enlace `a` con el atributo `href` para reedireccionar hacia `login.jsp` .

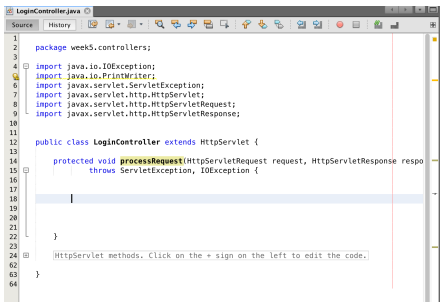
6.- La página `success.jsp` se modificará en las actividades posteriores. Por ahora se dejará el código que genera NetBeans por defecto.

Actividad 2 - Construcción de controladores

1.- Crear un servlet de nombre `LoginController` . Durante la creación del servlet deberá especificarse que sea creado dentro del paquete `week5.controllers` .



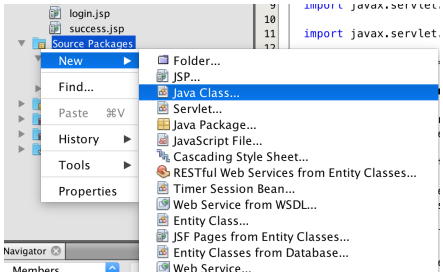
2.- Para inciar el trabajo con el código del servlet eliminaremos el contenido del método `processRequest`. De tal forma que quede como la siguiente imagen:

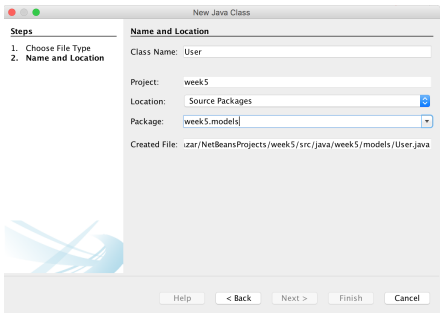


3.- Dentro del servlet, en el método `processRequest`, se deberá obtener los parámetros enviados por el cliente (usuario y contraseña) haciendo uso del método `request.getParameter`. El resto del código del servlet, se definirá en las siguientes actividades.

Actividad 3 - Contruccion de modelo

1.- Crear una clase de Java con nombre `User` que funcionará como modelo. La clase deberá estar dentro de un nuevo paquete de nombre `week5.models`, como aparece en la siguiente imagen:





2.- La clase `User` deberá contener lo siguiente:

- Dos propiedades privadas de tipo `String`: `username` y `password`.
- Un constructor que reciba como parámetro de entrada dos `String`. El primero representará el usuario y el segundo la contraseña. En el cuerpo del constructor deberá establecer el valor de los parámetros de entrada hacia las propiedades privadas `username` y `password`.
- Un método de nombre `getUsername` que devolverá un `String`, es decir, el valor de la variable privada `username`.

3.- Crear una segunda clase `Authentication`, dentro del mismo paquete `week5.models`, la cual deberá contener lo siguiente:

- Un método estático de nombre `authenticate` que devuelva un tipo `boolean`, el cual nos indicará si el usuario debió o no autenticarse.
- En el método `authenticate` definiremos dos variables de tipo `String`: `userDataBase` y `passwordDataBase` que actuarán como información *dummy*, es decir, información falsa y *hardcoded* para emular que obtuvimos esa información de la base de datos. Esta información *dummy*, nos servirá para comparar la información de "la base de datos" contra lo que capturó el usuario.
- La validación la realizaremos a través del método `equals`.
- El código siguiente muestra el ejemplo:

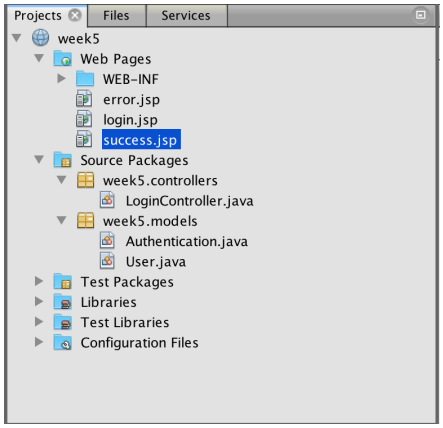
```
public class Authentication {

    public static boolean authenticate(String username, String password) {

        //Variables *dummy*.
        //Se hardcodean los valores "Miguel" y "MiPassword" para emular que se obtuvieron de una base de datos
        String userDataBase = "Miguel";
        String passwordDataBase = "MiPassword";

        //username.equals(userDataBase) realiza una comparación entre las cadenas username y userDataBase
        //Si son iguales devuelve true. Si son diferentes devuelve false.
        if(username.equals(userDataBase) && password.equals(passwordDataBase)) {
            return true;
        }
        else {
            return false;
        }
    }
}
```

Al finalizar las actividades anteriores, se deberá tener una estructura del proyecto como aparece en la siguiente imagen:



Actividad 4 - Refactorizacion de controladores y vistas.

1.- Una vez con las vistas, el controlador y los modelos, podemos continuar el código de nuestro controlador (servlet).

2.- En nuestro controlador, haremos uso de nuestros modelos. Para utilizarlos, debemos importar el paquete `week5.models.*`. Además, importaremos la clase `RequestDispatcher` que esta dentro del paquete `javax.servlet`:

```
import javax.servlet.RequestDispatcher;

import week5.models.User;
import week5.models.Authentication;
```

3.- Una vez importadas las clases necesarias dentro de nuestro controlador `LoginController`, construiremos las siguiente lógica en el método `processRequest`:

- Recuperar los valores enviados por el cliente, descrito en la Actividad 2.
- Invocar el método `authenticate` de la clase `Authentication`, descrito en la Actividad 3. Enviaremos la información de los valores obtenidos del `request`.
- Si el método `authenticate` devuelve `true` se deberá construir una instancia del modelo `User` y enviar un valor a la página `login.jsp`. Si el método devuelve `false` deberá redireccionar a la página `error.jsp`.
- El código siguiente muestra el ejemplo del método `processRequest`:

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

    //RequestDispatcher será la clase que nos apoyará a devolver
    //el valor hacia la vista
    RequestDispatcher dispatcher = null;

    //Se recupera la información del request
    String txtUsername = request.getParameter("txt-username");
    String txtPassword = request.getParameter("txt-password");

    //Invoca al método authenticate para validar el usuario
    boolean isValidUser =  Authentication.authenticate(txtUsername, txtPassword);

    if(isValidUser) {

        //Construye instancia del modelo User
        User user = new User(txtUsername, txtPassword);

        //Construye parámetro para enviar a la vista success.jsp
        request.setAttribute("username", user.getUsername());

        //Se define a donde se enviará el objeto request y response.
        dispatcher = request.getRequestDispatcher("success.jsp");
        dispatcher.forward(request, response);
    }
    else {
        //Si el usuario es inválido, reedireccionamos la petición hacia error.jsp
        response.sendRedirect("error.jsp");
    }
}
```

4.- Una vez creada la lógica del `LoginController`, ajustaremos la vista `success.jsp`. A continuación se muestra el código de ejemplo de la vista `success.jsp`:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Inicio de sesión válido</title>
    </head>
    <body>

        <h1>Inicio de sesión válido</h1>

        <h2>Hola <%= request.getAttribute("username") %></h2>
    </body>
</html>
```

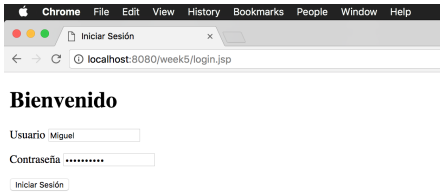
Actividad 5 - Prueba de la aplicación

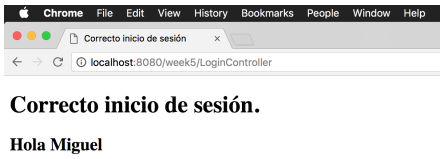
Primer prueba

1.- Ejecutar el proyecto y acceder a la url `http://localhost:8080/week5/login.jsp`

2.- Teclear el usuario y contraseña. Estos valores deberán coincidir con los descritos en la Actividad 2, las variables *dummy*.

3.- Dar clic en el botón Iniciar Sesión y validar que envía a la página de `succes.jsp` y además nos muestra un saludo con nuestro nombre.





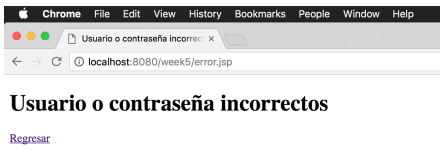
Segunda prueba

1.- Volver a ejecutar la página `login.jsp` .

2.- Teclear un usurio y contraseña incorrectos, es decir, que no coincidan con los descritos en las variables *dummy*.

3.- Dar clic en el botón Iniciar Sesión y validar que envía a la página de `error.jsp` .

4.- Probar el enlace "Regresar" y validar que reedirecciona hacia `login.jsp` .



Preguntas

- ¿Qué ventajas tienes al usar un patrón MVC en lugar de que un Servlet realice todo?
- Si quisieras agregar más vistas, ¿Qué cambios tendrías que hacer?
- Si requiriéramos más funcionalidades y no solamente un login de acceso, ¿Crees que un solo servlet o una sola clase de modelo serían suficientes?
- Crea un diagrama para este ejemplo basado en el diagrama habitual de MVC.

Especificaciones del Reporte

- El reporte debe incluir una portada con tus datos al inicio.
- El reporte debe contener la descripción de los pasos realizados para llevar a cabo la práctica del laboratorio. Cada paso debe contar con un fragmento de código o imagen que ilustre lo descrito. Piense en el reporte como una explicación para alguien ajeno al tema y detalle los puntos técnicos que sean necesarios.
- Contesta las preguntas mencionadas en la sección anterior.

- Comprime en un archivo `.zip` el directorio raíz de la práctica.
- El envío de la práctica debe incluir dos archivos: El reporte en `PDF` y el archivo `.zip` con el código fuente del proyecto.