UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE CIENCIAS FISICO MATEMATICAS

Diseño Orientado a Objetos

Practica Laboratorio 4

Maestro: Miguel Salazar

INTEGRANTE

Gilberto Alejandro Contreras Silva – 1683471

GRUPO: 00 AULA: 409

FECHA: Monterrey, N.L., 28 de febrero de 2017

Practica 4

En esta práctica creamos una página con la implementación de MVC. El objetivo de la práctica: Al finalizar la actividad deberás ser capaz de construir un sistema de autenticación simple basado en el patrón de diseño.

Primero que nada, creamos 3 páginas de vista, estas serán con formato Jsp, las cuales serán las de:

Success (cuando se haya podido logear correctamente)

Error (cuando el proceso de logeo falle)

Login (la que tiene el formulario e login)

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>COMPLETADO</title>
    </head>
    <body>
        <h1>Inicio de sesion extiso</h1>
        <h2>Hola <br><%= request.getAttribute("usuario") %></h2>
        <h2><%= request.getAttribute("email") %></h2>
        <h2><%= request.getAttribute("nombre") %></h2>
        <h2><%= request.getAttribute("apellidos") %></h2>
        <h2><%= request.getAttribute("ocupacion") %></h2>

    </body>
</html>
```

Esta es la página de Success, como veremos, tenemos que hacer uso de los request para que nos pueda imprimir los valores almacenados en esta propiedad del objeto.

El objeto es un tipo user, que está guardado en la sección de los models de este proyecto que más adelante lo explicare.

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>LOGIN</title>
    </head>
    <body>
        <h1>Registro</h1>
        <form action="LoginController" method="POST">
        Usuario:
        <input type="text" name="usuario">
        <br>
        Contraseña:
        <input type="password" name="password">
        <br>
        <input type="submit" value="Iniciar sesion">
        </form>
    </body>
</html>
```

Esta página es la del Login.jsp en la que podemos ver la creación de un formulario simple que solo pide usuario y contraseña para logearse. (Los demás datos deben de estar ya declarados con valores en el archivo de user.java)

```jsp
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>ERROR</title>
    </head>
    <body>
        <h1>USUARIO O CONTRASEÑA ERRONEA</h1>
        <a href="login.jsp">Regresar</a>
    </body>
</html>
```

Y finalmente este es de la página de error, la cual solo contiene el mensaje de que algo ocurrió mal y un hipervínculo para regresar al login.

Posteriormente pasamos a la creación de las clases de Java, las cuales serian 2, la clase que llevara a cabo la autenticación o la validación de los datos y la que contiene la clase del objeto principal, que en este caso es un usuario, con las propiedades o atributos de los datos que buscamos que imprima al final de la práctica.

Empezamos con el archivo de User

```java
public class user {
    private String usuario;
    private String password;
    String email;
    String nombre;
    String apellido;
    String ocupacion;


    public user() {
        this.usuario = "AzBlexx";
        this.password="contrasena";
        email= "azblexx@hotmail.com";
        nombre="Gilberto Alejanro";
        apellido="Contreras Silva";
        ocupacion="Estudiante";

    }
```

En esta parte del código, lo que hicimos fue crear la clase del objeto user y declarar las variables con almacenaran los datos que seamos.

Y abajo, el método constructor y la asignación de los valores.

Y posteriormente crear su get para que nos devuelva el valor. Esto se tiene que hacer con cada variable que tengamos.

```java
public String getUsuario() {
    return this.usuario;
}

public String getPassword(){
    return this.password;
}

public String getEmail(){
    return email;
}

public String getNombre(){
    return nombre;
}

public String getApellido(){
    return apellido;
}
```

Ahora sigue la autenticación:

```java
public class Authentication {
    public static boolean authenticate(String usuario, String password) {


        user user = new user();

        if(usuario.equals(user.getUsuario()) && password.equals(user.getPassword())) {
            return true;
        }
        else {
            return false;
        }
    }
```

Es esta parte solo comprobamos que los datos ingresados (no directamente, ya que en el controlador especificamos el nombre de la variable a comparar, pero eso lo veremos ahorita) con los datos almacenados en la clase user.

Finalmente el controlador.

```
package lab4.controllers;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.RequestDispatcher;
import lab4.models.Authentication;
import lab4.models.user;
```

Es importante no olvidar importar los documentos de los packets, porque de no ser asi, java no reconocerá las clases a las cuales referenciaremos mas adelante.

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

    RequestDispatcher dispatcher = null;

    String txtUsername = request.getParameter("usuario");
    String txtPassword = request.getParameter("password");

    boolean isValidUser =  Authentication.authenticate(txtUsername, txtPassword);

    if(isValidUser) {
    user user = new user();

    request.setAttribute("usuario", user.getUsuario());
    request.setAttribute("email", user.getEmail());
    request.setAttribute("nombre", user.getNombre());
    request.setAttribute("apellido", user.getApellido());
    request.setAttribute("ocupacion", user.getOcupacion());

    dispatcher = request.getRequestDispatcher("success.jsp");
```

Creamos una variable de tipo String para cada dato ingresado y la igualamos al valor del elemento que ingresamos (en este caso un input del formulario)

Después le damos esos valores a la autenticación que haremos y eso nos llevara al siguiente paso… que, de regresarme un valor verdadero, es decir, que los datos coincidan, se creara un nuevo objeto de tipo usuario y le asignaremos los valores con los métodos set.

```
    if(isValidUser) {
    user user = new user();

    request.setAttribute("usuario", user.getUsuario());
    request.setAttribute("email", user.getEmail());
    request.setAttribute("nombre", user.getNombre());
    request.setAttribute("apellido", user.getApellido());
    request.setAttribute("ocupacion", user.getOcupacion());

    dispatcher = request.getRequestDispatcher("success.jsp");
    dispatcher.forward(request, response);
}
else {

    response.sendRedirect("error.jsp");
}

}
```

Después la página nos redireccionara al success que creamos anteriormente y que como ahora ya tiene los datos, ya podemos hacer referencia a ellos para que los imprima directamente en la página.

En caso de que esta condición no se cumpla, es decir, que los datos no coincida, la página nos re direccionara a la página de error.

Preguntas de reportes

¿Qué ventajas tienes al usar un patrón MVC en lugar de que un Servlet realice todo?

Porque así tienes más organización a mi parecer, es decir, no está todo en un solo documento y no solo se maneja en el servlet. Aquí puedes crear clases abstractas, usar herencia, composición, valla es como si fuera un proyecto de java más elaborado que todo directamente desde el servlet. Eso sin mencionar los posibles problemas de seguridad.

Si quisieras agregar más vistas, ¿Qué cambios tendrías que hacer?

Crear los archivos JSP en su respectiva sesión, y si estas tienen un uso en particular tendría que referenciarlas en las demás.

Si requiriéramos más funcionalidades y no solamente un login de acceso, ¿Crees que un solo servlet o una sola clase de modelo serían suficientes?

No, ya que no estaríamos aprovechando todo el potencial de esto, esto puede hasta almacenar una base de datos completa y usar las clases de java haciendo todo lo que se pueda, no basta con una clase o con un solo servlet.