



Universidad Autónoma de Nuevo León
FACULTAD DE CIENCIAS FISICO MATEMATICAS



Laboratorio 4 Login

Alumno: Jose Alejandro Martinez Rivera

Matricula: 1680723 Grupo: 006

Maestro: Miguel Salazar

Monterrey, Nuevo León, martes 28 de Febrero de 2017

En esta practica realizamos un login, al finalizar la actividad pudimos ser capaces de construir un sistema de autenticación mediante el MVC.

Primeramente agregamos tres archivos de tipo JSP: login.jsp, success.jsp y error.jsp. Una vez hecho esto entramos a la página login.jsp y agregamos nuestro HTML que es la página que el cliente vera del lado del navegador web, ingresamos el código correspondiente para que al final diera un aspecto de la siguiente manera.



← → ↻ ⓘ localhost:8080/Laboratorio_4/login.jsp

Bienvenido

Usuario:

Contraseña:

Nombres:

Apellidos:

Email:

Ocupacion:

Después continuamos a modificar la página “error.jsp” la cual solo dará un mensaje al usuario que una contraseña que este incorrecta, además deberá contener un enlace que te direccionará de nuevo al a página principal que vendría siendo la de login. El codigo lo muestro a continuación:

```
6
7
8 <!--page contentType="text/html" pageEncoding="UTF-8"%-->
9 <!DOCTYPE html>
10 <html>
11 <head>
12 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
13 <title>Usuario incorrecto</title>
14 </head>
15 <body>
16 <h1>Usuario o contraseña incorrectas</h1>
17 <a href="login.jsp">Regresar</a>
18
19 </body>
20 </html>
21
22
```

Y el aspecto visual para el cliente será de la siguiente manera:



← → ↻ ⓘ localhost:8080/Laboratorio_4/error.jsp

Usuario o contraseña incorrectas

[Regresar](#)

Construcción de controladores

Ahora deberemos crear el servlet llamado LoginController, que será creado dentro del paquete week5.controllers, cuando sea creado el servlet deberemos eliminar parte del código predeterminado.

Aquí obtendremos los parámetros mediante el método request.getParameter, y esos parámetros se los asignamos a unas variables creadas:

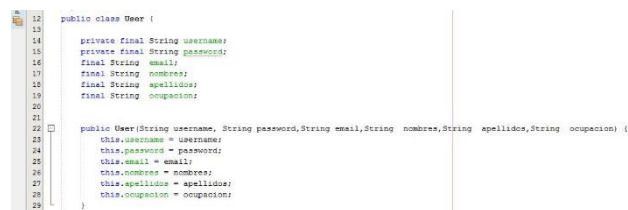


```
33 protected void processRequest(HttpServletRequest request, HttpServlet
34     throws ServletException, IOException {
35
36
37
38     RequestDispatcher dispatcher = null;
39
40
41     String username = request.getParameter("username");
42     String password = request.getParameter("password");
43     String nombres = request.getParameter("nombres");
44     String apellidos = request.getParameter("apellidos");
45     String email = request.getParameter("email");
46     String ocupacion = request.getParameter("ocupacion");
47
```

Construcción de modelo

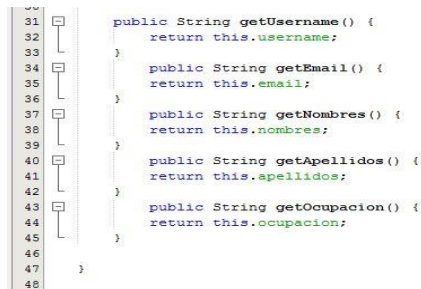
Creamos una clase de Java con el nombre de User, esta deberá ser creada dentro de un paquete llamado week5.models.

En esta clase primero declaramos las variables privadas “username” y “password”, así mismo agregamos los demás parámetros a utilizar. Debajo de esto creamos el constructor donde estableceremos los parámetros hacia las propiedades a utilizar, el código lo muestro a continuación:



```
12 public class User {
13
14     private final String username;
15     private final String password;
16     final String email;
17     final String nombres;
18     final String apellidos;
19     final String ocupacion;
20
21
22     public User(String username, String password, String email, String nombres, String apellidos, String ocupacion) {
23         this.username = username;
24         this.password = password;
25         this.email = email;
26         this.nombres = nombres;
27         this.apellidos = apellidos;
28         this.ocupacion = ocupacion;
29     }
30
```

Y después crearemos un método de nombre get“nombre de especificacion a obtener” que devolverá un String, hacia las variables creadas anteriormente, respectivamente.



```
31 public String getUsername() {
32     return this.username;
33 }
34 public String getEmail() {
35     return this.email;
36 }
37 public String getNombres() {
38     return this.nombres;
39 }
40 public String getApellidos() {
41     return this.apellidos;
42 }
43 public String getOcupacion() {
44     return this.ocupacion;
45 }
46
47 }
48
```

Una vez hecho lo anterior, crearemos una nueva clase de Java llamada Authentication en la cual crearemos el método authenticate , que nos servirá para validar que el usuario y contraseña que se esté ingresando sea válida, comparando estos valores con un database creado ahí mismo.

El código es el siguiente:

```
11  L  */
12  public class Authentication {
13
14  public static boolean authenticate(String username, String password) {
15
16      String userDataBase = "Alex";
17      String passwordDataBase = "hola";
18
19      if(username.equals(userDataBase) && password.equals(passwordDataBase)){
20          return true;
21      }
22      else {
23          return false;
24      }
25  }
```

Regresamos al servlet, donde importaremos carpertas encontradas en paquete javax.servlet. Dentro del método processRequest recuperaremos los valores enviados por el cliente, dentro de este mismo invocaremos el método authenticate de la clase Authentication, donde si el método devuelve un true se deberá construir una instancia del método user y enviar un valor a la página logging.jsp y si devuelve un false nos re direccionaremos a la página error.

El código que utilizamos para crear toda esa lógica es el siguiente:

```
47
48  boolean isValidUser = Authentication.authenticate(username,password);
49
50  if(isValidUser) {
51
52
53      User user = new User(username, password, nombres, email,apellidos,ocupacion);
54
55
56      request.setAttribute("username", user.getUsername());
57      request.setAttribute("email", user.getEmail());
58      request.setAttribute("nombres", user.getNombres());
59      request.setAttribute("apellidos", user.getApellidos());
60      request.setAttribute("ocupacion", user.getOcupacion());
61
62
63      dispatcher = request.getRequestDispatcher("success.jsp");
64      dispatcher.forward(request, response);
65  }
66  else {
67
68      response.sendRedirect("error.jsp");
69  }
70
```

Ya por ultimo nos quedara mostrar el mensaje que se contestara, cuando se valide el usuario y contraseña.

Página success

Regresamos a la página succes.jsp le damos el título de Inicio de sesión valido, agregamos varios h2 con un poco de sintaxis de JSP donde ponemos el método request.getAttribute("") y dentro del paréntesis el parámetro a imprimir.

A continuación, el código:

```
3      Created on : 26/02/2017, 09:12:19 PM
4      Author    : JoseAlejandro
5      -->
6
7      <%@page contentType="text/html" pageEncoding="UTF-8"%>
8      <!DOCTYPE html>
9      <html>
10     <head>
11     <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
12     <title>Inicio de sesion valido</title>
13     </head>
14     <body>
15     <h1>Inicio de sesion valido </h1>
16     <h2>Hola <%= request.getAttribute("username") %></h2>
17     <h2>Email:<%= request.getAttribute("email") %></h2>
18     <h2>Nombres:<%= request.getAttribute("nombres") %></h2>
19     <h2>Apellidos: <%= request.getAttribute("apellidos") %></h2>
20     <h2>Ocupacion:<%= request.getAttribute("ocupacion") %></h2>
21     </body>
22     </html>
```

Una vez realizado lo anterior y ya con la integración de las clases de java y las demás paginas obtenemos el siguiente resultado cuando el usuario si esta validado:



NOTA: Anteriormente ya había mostrado el resultado en caso de usuario o contraseñas incorrectos.

Reflexión

El uso de un MVC tiene muchas ventajas, entre las cuales es el poder dividir la lógica del diseño haciendo el proyecto más escalable, lo anterior el profesor nos lo recalca mucho y después de investigar un tiempo me di cuenta que muchos concuerdan con la escalabilidad que se le puede dar un proyecto.

Así mismo podemos utilizar métodos comunes de programación, esto hace que el código sea más entendible entre estos, pudiendo uno continuar el trabajo de otro.

Pienso que podemos agregar más funcionalidades con un solo servlet, sin la necesidad de hacer más o más modelos, aunque pienso que para tener una mejor organización de nuestro proyecto debemos nombrar bien nuestros Servlets o clases para poder hacer buen uso de todas las funcionalidades a agregar.

Diagrama MVC

