

**Tipo** : Guía de laboratorio  
**Capítulo** : Spring MVC  
**Duración** : 80 minutos

---

## I. OBJETIVO

Crear un login de usuario utilizando Data Binding.

## II. REQUISITOS

Los siguientes elementos de software son necesarios para la realización del laboratorio.

- NetBeans 11
- JDK 11

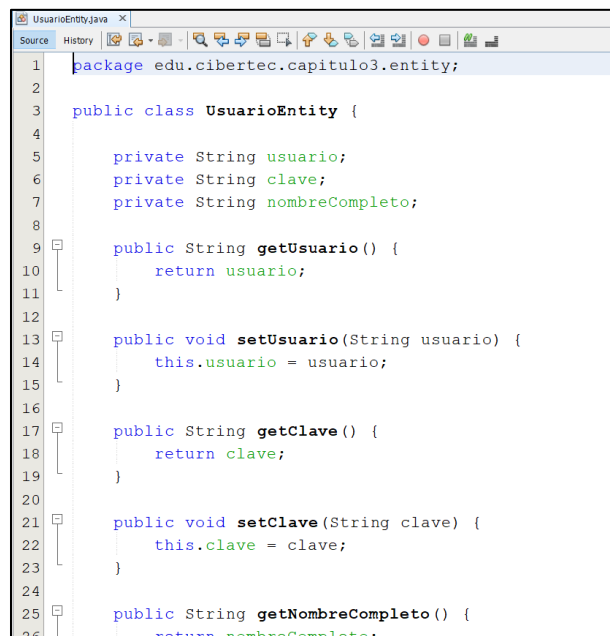
## III. EJECUCIÓN DEL LABORATORIO

### Ejercicio:

Modifica el proyecto para crear una pantalla de login donde se verifique los valores ingresados, se utilicen diferentes capas de aplicación y los datos sean llenados mediante Data Binding.

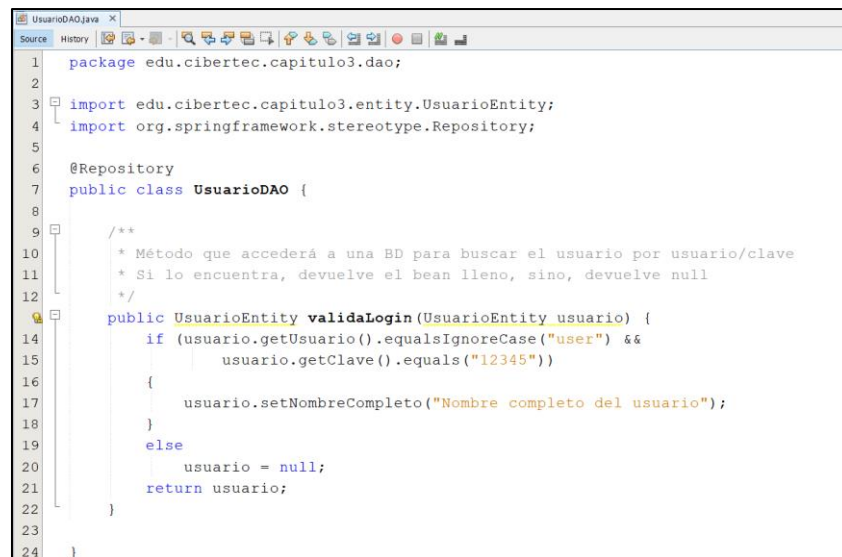
1. Empieza el laboratorio creando las partes de abajo hacia arriba. Primero, diseña una clase que represente la tabla de usuario, contando con 3 campos: usuario, clave y nombre completo.

Para realizar esta operación elabora la clase “**UsuarioEntity**” dentro del paquete “**entity**” de la forma.



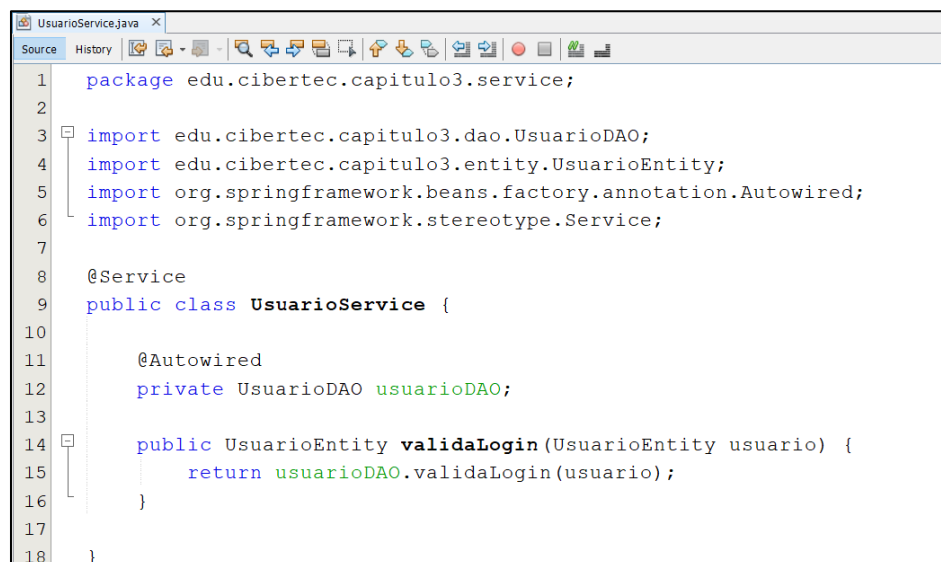
```
1 package edu.cibertec.capitulo3.entity;
2
3 public class UsuarioEntity {
4
5     private String usuario;
6     private String clave;
7     private String nombreCompleto;
8
9     public String getUsuario() {
10         return usuario;
11     }
12
13     public void setUsuario(String usuario) {
14         this.usuario = usuario;
15     }
16
17     public String getClave() {
18         return clave;
19     }
20
21     public void setClave(String clave) {
22         this.clave = clave;
23     }
24
25     public String getNombreCompleto() {
26         return nombreCompleto;
27     }
28 }
```

2. La siguiente capa a crear es la DAO. Diseña la clase UsuarioDAO dentro del paquete “dao”, la cual solo contiene un método para la validación y obtención del nombre completo del usuario que se está logueando. La implementación es de la siguiente manera.



```
1 package edu.cibertec.capitulo3.dao;
2
3 import edu.cibertec.capitulo3.entity.UsuarioEntity;
4 import org.springframework.stereotype.Repository;
5
6 @Repository
7 public class UsuarioDAO {
8
9     /**
10      * Método que accederá a una BD para buscar el usuario por usuario/clave
11      * Si lo encuentra, devuelve el bean lleno, sino, devuelve null
12      */
13
14     public UsuarioEntity validaLogin(UsuarioEntity usuario) {
15         if (usuario.getUsuario().equalsIgnoreCase("user") &&
16             usuario.getClave().equals("12345"))
17         {
18             usuario.setNombreCompleto("Nombre completo del usuario");
19         }
20         else
21             usuario = null;
22         return usuario;
23     }
24 }
```

3. La siguiente capa para crear es la de servicio, la cual invoca la capa DAO y contiene un solo método bastante sencillo. La clase se le denomina “UsuarioService” y está dentro del paquete “service”. La implementación es como la siguiente.



```
1 package edu.cibertec.capitulo3.service;
2
3 import edu.cibertec.capitulo3.dao.UsuarioDAO;
4 import edu.cibertec.capitulo3.entity.UsuarioEntity;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 @Service
9 public class UsuarioService {
10
11     @Autowired
12     private UsuarioDAO usuarioDAO;
13
14     public UsuarioEntity validaLogin(UsuarioEntity usuario) {
15         return usuarioDAO.validaLogin(usuario);
16     }
17
18 }
```

4. La siguiente capa es el Controlador. En esta capa crea la clase que obtiene los parámetros desde el formulario y que invoca al servicio para poder obtener la lógica de negocio de la aplicación.

El controlador tiene 2 métodos, el primero para mostrar el formulario a llenar y otro para recibir el “action” del formulario de login.

Crea la clase “**UsuarioController**” dentro del paquete “**controller**” y la implementas de la siguiente manera.

```
package edu.cibertec.capitulo3.controller;

import edu.cibertec.capitulo3.entity.UsuarioEntity;
import edu.cibertec.capitulo3.service.UsuarioService;
import javax.servlet.http.HttpServletRequest;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class UsuarioController {

    @Autowired
    private UsuarioService usuarioService;

    @RequestMapping("loginMostrar")
    public String loginMostrar() {
        return "login";
    }

    @RequestMapping("loginAccion")
    public ModelAndView loginAccion(HttpServletRequest request) {
        ModelAndView mv = null;
        UsuarioEntity usuarioValida = new UsuarioEntity();
        usuarioValida.setUsuario(request.getParameter("txtUsuario"));
        usuarioValida.setClave(request.getParameter("txtClave"));

        UsuarioEntity ue = usuarioService.validaLogin(usuarioValida);
        if (ue == null) {
            mv = new ModelAndView("login", "msgError", "Usuario y clave no existen. Vuelva a intentar!");
        } else {
            mv = new ModelAndView("saludo", "mensaje", "Bienvenido " + ue.getNombreCompleto());
        }
        return mv;
    }
}
```

5. Diseña la página de login, donde se tienen 2 cajas de texto y el botón “**submit**”. La página se llama “**login.jsp**” y debe estar localizada en el folder “**WEB-INF/views**”. La implementación es como se muestra a continuación (se realiza sin estilos para acortar el código, después se pondrá).

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Login de Usuario</title>
  </head>
  <body>
    <h1>Login de usuario</h1>
    <br/>
    <br/>
    <form action="loginAccion.htm" method="post">
      Usuario: <input type="text" name="txtUsuario">
      <br/>
      Clave: <input type="password" name="txtClave">
      <br/>
      <input type="submit" value="Ingresar" />
    </form>
    <% if (request.getAttribute("msgError")!=null) {
      out.print(request.getAttribute("msgError"));
    } %>
  </body>
</html>
```

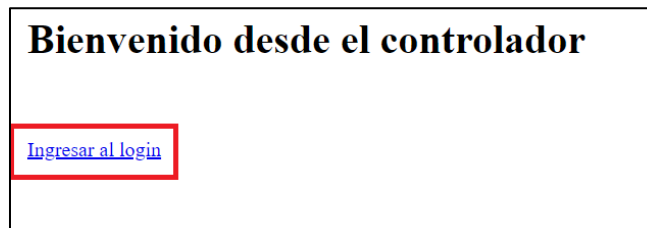
6. Como paso final, elabora un enlace desde la página de “**saludo**” para que invoque la página de login y empiece el flujo del ejercicio.

Modifica el archivo “**saludo.jsp**” adicionando la siguiente línea.

```
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Primera Aplicación Spring MVC</title>
</head>
<body>
  <h1>${mensaje}</h1>
  <br /><br />
  <a href="loginMostrar.htm">Ingresar al login</a>
</body>
```

7. Ejecuta la aplicación y sigue los pasos.

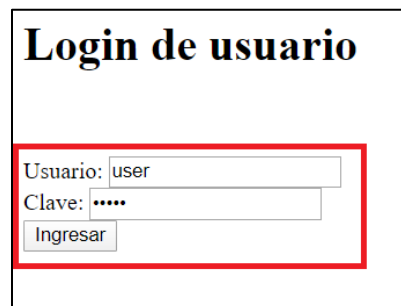
Selecciona el enlace de la primera página.



**Bienvenido desde el controlador**

[Ingresar al login](#)

Ingresa “user” y “12345” en las cajas y presiona sobre el botón “Ingresar”.



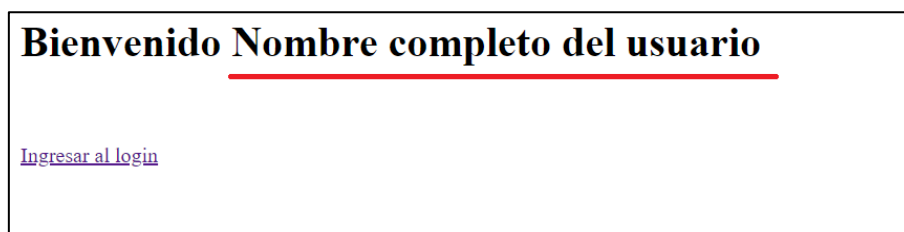
**Login de usuario**

Usuario: user

Clave: .....

Ingresar

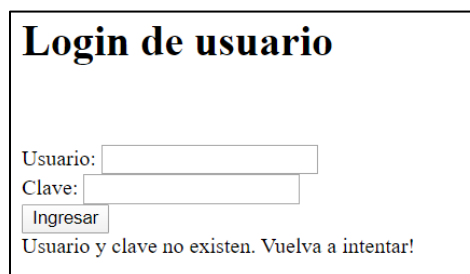
Se muestra la página con el nombre completo del usuario.



**Bienvenido Nombre completo del usuario**

[Ingresar al login](#)

Ahora, prueba si el usuario y clave no son correctos. Haz clic nuevamente al enlace de login y procede a ingresar un usuario o clave diferente para ver el mensaje de error.



**Login de usuario**

Usuario:

Clave:

Ingresar

Usuario y clave no existen. Vuelva a intentar!

### Usar DataBinding directo

Hasta ahora, en el ejercicio no se ha usado DataBinding, la captura de los campos del JSP se ha realizado mediante un `request.getParameter`, asimismo la creación y llenado del bean de destino se realizó manualmente en el Controller. Ahora cambiaremos el ejercicio para adecuarlo a DataBinding

1. Como primer paso, cambia el formulario para que los campos que se envían al Servlet tengan los mismos nombres de las propiedades del Bean.

```
<body>
  <h1>Login de usuario</h1>
  <br/>
  <br/>
  <form action="loginAccion.htm" method="post">
    Usuario: <input type="text" name="usuario">
    <br/>
    Clave: <input type="password" name="clave">
    <br/>
    <input type="submit" value="Ingresar" />
  </form>
  <% if (request.getAttribute("msgError") != null) {
    out.print(request.getAttribute("msgError"));
  } %>
</body>
```

2. Lo siguiente será cambiar el Controller para indicarle al framework cuál será el Bean que tiene que llenar y cambiar la lógica pues ya el bean se llena automáticamente.

```
@RequestMapping("loginMostrar")
public String loginMostrar() {
    return "login";
}

@RequestMapping("loginAccion")
public ModelAndView loginAccion(UsuarioEntity usuarioValida) {
    ModelAndView mv = null;

    UsuarioEntity ue = usuarioService.validaLogin(usuarioValida);
    if (ue == null) {
        mv = new ModelAndView("login", "msgError", "Usuario y clave no existen. Vuelva a intentar!");
    } else {
        mv = new ModelAndView("saludo", "mensaje", "Bienvenido " + ue.getNombreCompleto());
    }
    return mv;
}
```

3. Ejecuta la aplicación y sigue obteniendo la respuesta anterior sin ningún cambio.

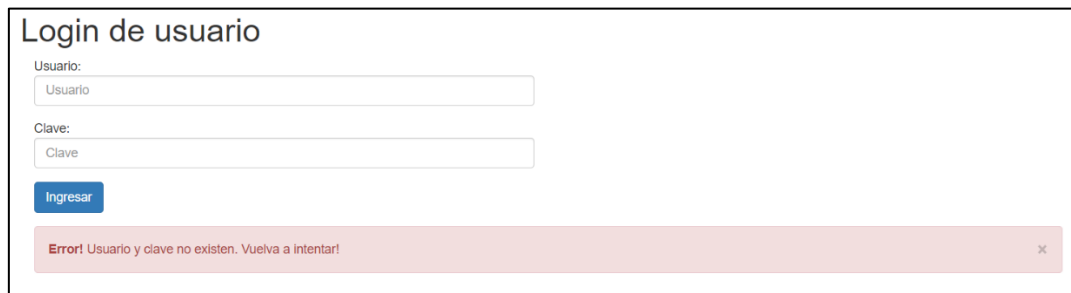
**Bienvenido Nombre completo del usuario**

[Ingresar al login](#)

4. Modifica el login.jsp para colocar estilos con Bootstrap.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
    <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
  >
    <title>Login de Usuario</title>
  </head>
  <body>
    <br/>
    <div class="container">
      <div class="row d-flex justify-content-center mx-auto">
        <h1>Login de usuario</h1>
        <div class="col-md-6 col-xs-12 div-style">
          <form action="loginAccion.htm" method="post">
            <div class="form-group">
              Usuario: <input type="text" name="usuario" class="form-control text-
box"
              required placeholder="Usuario">
            </div>
            <div class="form-group">
              Clave: <input type="password" name="clave" class="form-control text-
box"
              required placeholder="Clave">
            </div>
            <div class="form-group justify-content-center d-flex">
              <input type="submit" value="Ingresar" class="btn btn-primary button-
submit">
            </div>
          </form>
        </div>
      </div>
      <% if (request.getAttribute("msgError") != null) {%>
        <div class="alert alert-danger">
          <strong>Error!</strong> <%= request.getAttribute("msgError")%>
          <button type="button" class="close" data-dismiss="alert">&times;</button>
        </div>
      <% }%>
    </div>
  </body>
</html>
```

5. Ejecuta la aplicación y el login debe verse de la siguiente manera.



**Login de usuario**

Usuario:

Clave:

**Error!** Usuario y clave no existen. Vuelva a intentar! ×

#### IV. EVALUACIÓN

**Añade las siguientes validaciones al ejercicio.**

1. Que el usuario tenga por lo menos 3 caracteres de longitud.
2. Que la clave tenga por lo menos un dígito.
3. Que la clave y el usuario no sean iguales.