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

I. OBJETIVO

Crear el mantenimiento del usuario utilizando Form Tag Library.

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 el mantenimiento de la "tabla" de usuarios utilizando la librería de Tag de Spring.

1. Empieza el laboratorio modificando el entity de Usuario para crear un constructor donde se puede inicializar sus propiedades en una sola línea y también el constructor por defecto. De igual manera, aumenta las siguientes líneas al archivo "**UsuarioEntity**".

   *NOTA: esto se puede realizar mediante la ayuda del IDE, para ello haz clic derecho sobre el código y selecciona "**Insert Code**" -> "**Constructor**".*

```
public class UsuarioEntity {

    private String usuario;
    private String clave;
    private String nombreCompleto;

    public UsuarioEntity() {
    }

    public UsuarioEntity(String usuario, String clave, String nombreCompleto) {
        this.usuario = usuario;
        this.clave = clave;
        this.nombreCompleto = nombreCompleto;
    }

    public String getUsuario() {
        return usuario;
    }

    public void setUsuario(String usuario) {
        this.usuario = usuario;
    }
```

2. En la capa DAO crea una lista de usuarios "**predeterminada**" (para fingir una Base de datos) y un método de creación de Usuario.

```
@Repository
public class UsuarioDAO {

    private static List<UsuarioEntity> listaUsuarios;

    static {
        listaUsuarios = new ArrayList<>();
        listaUsuarios.add(new UsuarioEntity("jose", "12345", "José Perez"));
        listaUsuarios.add(new UsuarioEntity("maria", "54321", "María Quispe"));
        listaUsuarios.add(new UsuarioEntity("yaddif", "58475", "Yaddif Medina"));
        listaUsuarios.add(new UsuarioEntity("olga", "po54d", "Olga Ibarra"));
    }

    public void insertaUsuario(UsuarioEntity ue) {
        listaUsuarios.add(ue);
    }

    public List<UsuarioEntity> getListaUsuarios() {
        return listaUsuarios;
    }
```

3. Modifica la capa de servicio para que invoque a los dos métodos nuevos creados en el DAO.

```
@Service
public class UsuarioService {

    @Autowired
    private UsuarioDAO usuarioDAO;

    public UsuarioEntity validaLogin(UsuarioEntity usuario) {
        return usuarioDAO.validaLogin(usuario);
    }

    public void insertaUsuario(UsuarioEntity ue) {
        usuarioDAO.insertaUsuario(ue);
    }

    public List<UsuarioEntity> getListaUsuarios() {
        return usuarioDAO.getListaUsuarios();
    }
}
```

4. Modifica el Controlador para que después de validar el usuario del login llame a la página de mantenimiento inicial que permita mostrar la lista de usuarios. El Controller simplemente recuperará la lista de usuarios y se la enviará a la página para que sea mostrada.

```
@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("usuarioLista", "lista", usuarioService.getListaUsuarios());
    }
    return mv;
}
```

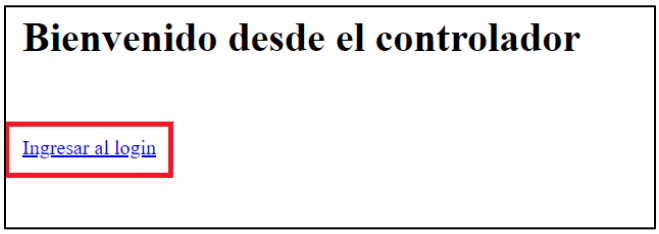
5. Ahora diseña la página inicial del mantenimiento del usuario llamada "**usuarioLista.jsp**" donde se mostrará la lista de usuario y se tendrá el botón de insertar. Los estilos se colocarán después.

```
<%@page import="edu.cibertec.capitulo3.entity.UsuarioEntity"%>
<%@page import="java.util.List"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Mantenimiento de usuarios</title>
  </head>
  <body>
    <h1>Listado de Usuarios</h1>
    <br/>
    <% List<UsuarioEntity> lista = (List<UsuarioEntity>)request.getAttribute("lista");
%>
    <table border="1">
      <thead>
        <tr>
          <th>Usuario</th>
          <th>Clave</th>
          <th>Nombre completo</th>
        </tr>
      </thead>
      <tbody>
        <% for (UsuarioEntity usuario : lista) { %>
        <tr>
          <td><%= usuario.getUsuario() %></td>
          <td><%= usuario.getClave()%></td>
          <td><%= usuario.getNombreCompleto()%></td>
        </tr>
        <% } %>
```

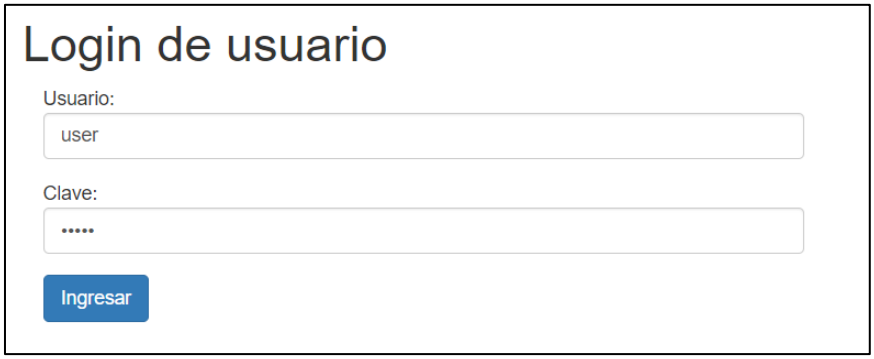
```
        </tbody>
      </table>
    </body>
</html>
```

6. Ejecuta la aplicación y sigue los pasos.

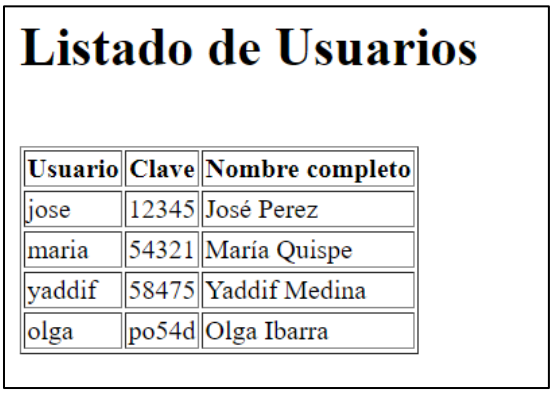
   Selecciona el enlace de la primera página.



   Ingresa "**user"** y "**12345**" en las cajas y presiona el botón "**Ingresar**".



   Se muestra la página con el listado de los usuarios.



7. El siguiente paso es crear la pantalla de inserción de usuarios, el controlador que lo maneje y las validaciones en el bean para asegurar los datos a ingresar.

   Inicia elaborando la página de inserción. Coloca el nombre de la página "**usuarioDatos.jsp**" e implementa de la siguiente manera (la parte de estilos se verá después).

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="mvc" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Datos del Usuario</title>
    <style type="text/css">
      .formFieldError { background-color: #FFC; }
    </style>
  </head>
  <body>
    <h1>Datos del usuario</h1>
    <br />
    <mvc:form modelAttribute="usuarioBean" action="usuarioGrabar.htm">
      <table>
        <tr>
          <td><mvc:label  path="usuario">Usuario: </mvc:label></td>
          <td><mvc:input  path="usuario" cssErrorClass="formFieldError" /></td>
          <td><mvc:errors path="usuario" /></td>
        </tr>
        <tr>
          <td><mvc:label  path="clave">Clave: </mvc:label></td>
          <td><mvc:input  path="clave" cssErrorClass="formFieldError" /></td>
          <td><mvc:errors path="clave" /></td>
        </tr>
        <tr>
          <td><mvc:label  path="nombreCompleto">Nombre Completo:
</mvc:label></td>
          <td><mvc:input  path="nombreCompleto" cssErrorClass="formFieldError"
/></td>
          <td><mvc:errors path="nombreCompleto" /></td>
        </tr>
        <tr>
          <td colspan="1">
            <input type="submit" value="Insertar" />
          </td>
        </tr>
      </table>
    </mvc:form>
  </body>
</html>
```

8. Crea un enlace para invocar desde la lista a la página de crear usuario.

```
                    <th>Nombre completo</th>
                </tr>
            </thead>
            <tbody>
                <% for (UsuarioEntity usuario : lista) {%>
                <tr>
                    <td><%= usuario.getUsuario()%></td>
                    <td><%= usuario.getClave()%></td>
                    <td><%= usuario.getNombreCompleto()%></td>
                </tr>
                <% }%>
            </tbody>
        </table>
        <a href="usuarioCrear.htm">Crear Usuario</a>
    </body>
</html>
```

9. Modifica el Controller para tener 2 métodos adicionales, el primero para realizar la llamada a la página de creación y el segundo para la grabación de los datos que se puede enviar desde esta.

```
@RequestMapping("usuarioCrear")
public ModelAndView usuarioCrear(){
    return new ModelAndView("usuarioDatos", "usuarioBean", new UsuarioEntity());
}

@RequestMapping("usuarioGrabar")
public ModelAndView usuarioGrabar(@Valid @ModelAttribute("usuarioBean")
        UsuarioEntity usuarioValida, BindingResult result, ModelMap modelMap) {

    ModelAndView mv = null;

    if (result.hasErrors())
    {
        mv = new ModelAndView("usuarioDatos", "usuarioBean", usuarioValida);
    }
    else
    {
        usuarioService.insertaUsuario(usuarioValida);
        mv = new ModelAndView("usuarioLista", "lista", usuarioService.getListaUsuarios());
    }
    return mv;
}
```

10. Para realizar la validación de los campos directamente, agrega la librería de Hibernate en Maven para que sea integrada al proyecto.

```
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.1.11.RELEASE</version>
        <scope>compile</scope>
    </dependency>
    <dependency>
        <groupId>org.hibernate</groupId>
        <artifactId>hibernate-validator</artifactId>
        <version>5.1.3.Final</version>
    </dependency>
    <dependency>
        <groupId>javax.xml.bind</groupId>
        <artifactId>jaxb-api</artifactId>
        <version>2.3.0</version>
    </dependency>
dependencies>
```

11. Modifica el entity para colocarle validaciones a los campos que se deseen. Coloca algunos ejemplos como los siguientes.

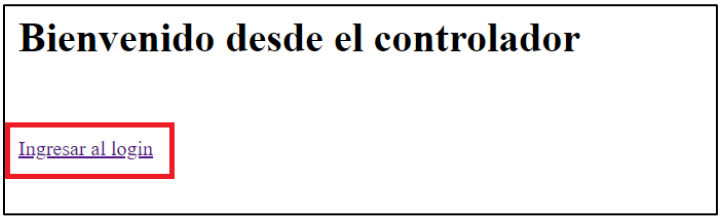
```
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import org.hibernate.validator.constraints.NotBlank;

public class UsuarioEntity {

    @Size(min=3, max=20)
    private String usuario;

    @NotNull
    @NotBlank
    private String clave;
    private String nombreCompleto;

    public UsuarioEntity() {
    }

    public UsuarioEntity(String usuario, String clave, String nombreCompleto) {
        this.usuario = usuario;
        this.clave = clave;
        this.nombreCompleto = nombreCompleto;
    }
```

12. El último cambio será indicarle al frameqork que acepte el manejo de anotaciones para las funciones del MVC como tal. Esto lo realizas ingresando la siguiente línea en el archivo de configuración del Dispatcher.

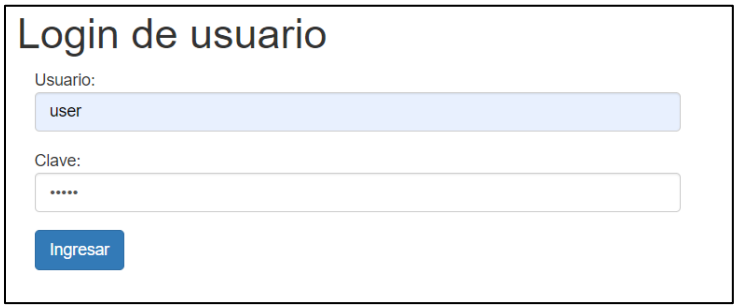
```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd">

    <context:component-scan base-package="edu.cibertec.capitulo3" />
    <mvc:annotation-driven />

    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/views/" />
        <property name="suffix" value=".jsp" />
    </bean>

</beans>
```

13. Ejecuta la aplicación y sigue los siguientes pasos.

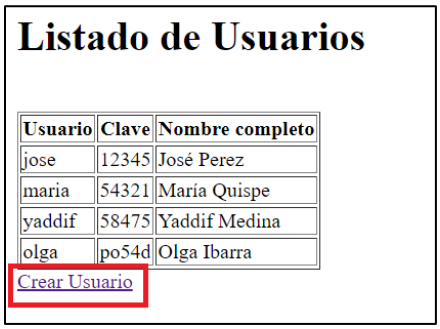
Haz clic para ingresar al login.



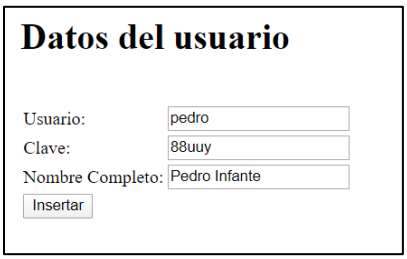
Ingresa el usuario y clave de manera correcta.



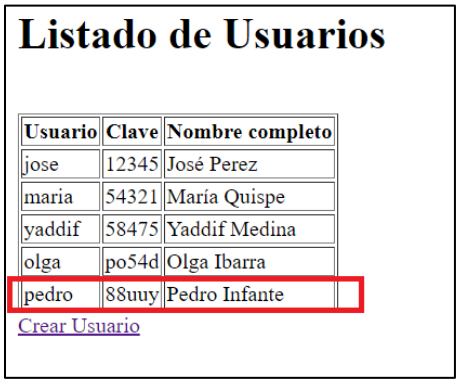
Haz clic en el enlace para la inserción.



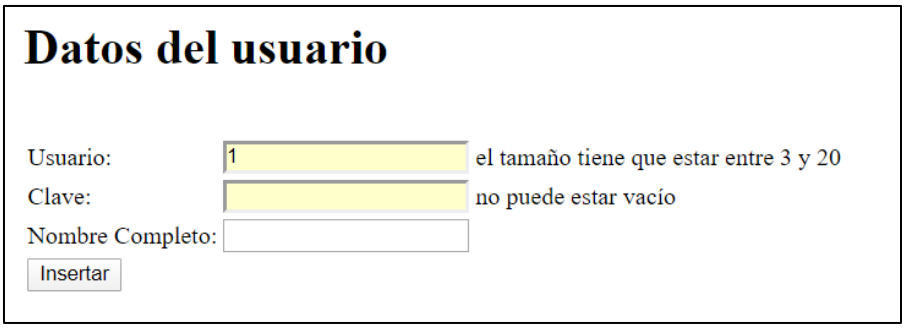
Ingresa datos en los campos y presiona sobre el botón "**Insertar**".



Los datos ingresados fueron “**grabados**” en la lista y ahora se muestran.



Vuelve a dar clic en el enlace de “**Crear usuario**”, ingresa solo un carácter en el campo usuario y presiona sobre el botón “**insertar**” para ver la validación de los datos.



14. Una vez que el ejemplo ejecuta sin problemas, procede a colocar estilos. Empieza con la página “**usuarioLista.jsp**”.

```
<%@page import="edu.cibertec.capitulo3.entity.UsuarioEntity"%>
<%@page import="java.util.List"%>
<%@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>Mantenimiento de usuarios</title>
  </head>
  <body>
    <h1>Listado de Usuarios</h1>
    <br/>
    <% List<UsuarioEntity> lista = (List<UsuarioEntity>) request.getAttribute("lista");
%>
    <div class="table-responsive">
```

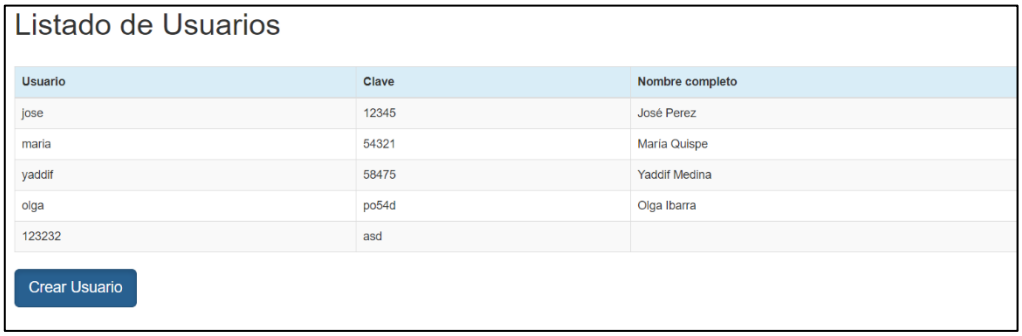
```
<table class="table table-responsive table-sm table-dark table-striped table-bordered
table-hover">
        <thead>
          <tr class="info">
            <th>Usuario</th>
            <th>Clave</th>
            <th>Nombre completo</th>
          </tr>
        </thead>
        <tbody>
          <% for (UsuarioEntity usuario:lista) {%>
          <tr>
            <td><%= usuario.getUsuario()%></td>
            <td><%= usuario.getClave()%></td>
            <td><%= usuario.getNombreCompleto()%></td>
          </tr>
          <% }%>
        </tbody>
      </table>
      </div>
      <a href="usuarioCrear.htm" class="btn btn-primary btn-lg active">Crear
Usuario</a>
   </body>
</html>
```

15. También en la página de datos (“usuarioDatos.jsp”).

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib uri="http://www.springframework.org/tags/form" prefix="mvc" %>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Datos del Usuario</title>
      <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
>
      <style type="text/css">
         .formFieldError { background-color: #FFC; }
      </style>
   </head>
   <body>
      <div class="container">
         <div class="row d-flex justify-content-center mx-auto">

            <h1>Datos del usuario</h1>
            <div class="col-md-6 col-xs-12 div-style">
```

```
            <mvc:form modelAttribute="usuarioBean" action="usuarioGrabar.htm">
               <table>
                  <tr>
                     <td><mvc:label  path="usuario">Usuario: </mvc:label></td>
                     <td><mvc:input  path="usuario" cssErrorClass="form-control text-
box formFieldError"
                              cssClass="form-control text-box" /></td>
                     <td><mvc:errors path="usuario" /></td>
                  </tr>
                  <tr>
                     <td><mvc:label  path="clave">Clave: </mvc:label></td>
                     <td><mvc:input  path="clave" cssErrorClass="form-control text-box
formFieldError"
                              cssClass="form-control text-box"/></td>
                     <td><mvc:errors path="clave" /></td>
                  </tr>
                  <tr>
                     <td><mvc:label  path="nombreCompleto">Nombre Completo:
</mvc:label></td>
                     <td><mvc:input  path="nombreCompleto" cssErrorClass="form-
control text-box formFieldError"
                              cssClass="form-control text-box"/></td>
                     <td><mvc:errors path="nombreCompleto" /></td>
                  </tr>
                  <tr>
                     <td colspan="1">
                        <input type="submit" value="Insertar" class="btn btn-primary
button-submit"/>
                     </td>
                  </tr>
               </table>
            </mvc:form>
         </div>
      </div>
   </div>
  </body>
</html>
```

Ejecuta la aplicación y observa los cambios de las páginas.



IV. EVALUACIÓN

Añade lo siguiente al ejercicio.

1. La edición y eliminación de usuarios.
2. Botón "**Regresar**" desde la página de datos para cuando el usuario no quiera realizar la acción.