Tipo : Guía de laboratorio
Capítulo : Spring MVC
Duración : 60 minutos

I. OBJETIVO

Añadir al mantenimiento de usuario el colocarle una foto y un contador de los usuarios creados desde que se logueó mediante el uso de sesiones.

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 poder tener la foto del usuario y poder contabilizar las veces que se ha creado registros desde que ingresó al sistema.

1. Empieza el laboratorio modificando el entity de Usuario para crear una propiedad donde almacenarás la foto, esta propiedad debe ser del tipo arreglo de bytes. Luego, modifica la clase **UsuarioEntity** con el incremento de la propiedad y sus métodos get/set.

```
@NotNull
@NotBlank
private String clave;
private String nombreCompleto;
private byte[] foto;

public byte[] getFoto() {
    return foto;
}

public void setFoto(byte[] foto) {
    this.foto = foto;
}

public UsuarioEntity() {
}

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

2. Adiciona un método en la clase DAO para que mediante el código de usuario se pueda devolver el objeto Entity completo.

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

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

public UsuarioEntity getUsuario(String codigo) {
    UsuarioEntity rpta = null;
    for (UsuarioEntity usuario : listaUsuarios) {
        if (usuario.getUsuario().equalsIgnoreCase(codigo))
        {
            rpta = usuario;
            break;
        }
    }
    return rpta;
}
```

3. En la capa de servicio también crea el método para la búsqueda de un solo usuario por su código.

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

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

public UsuarioEntity getUsuario(String codigo) {
    return usuarioDAO.getUsuario(codigo);
}
```

4. Lo siguiente es modificar el Controller añadiendo los siguientes métodos.

```
  @RequestMapping("fotoMostrar")
  public String fotoMostrar(HttpServletRequest request, Model modelo) {
     modelo.addAttribute("usuario",
usuarioService.getUsuario(request.getParameter("codigoUsuario")));
     return "fotoUsuario";
  }

  @RequestMapping("fotoGrabar")
  public ModelAndView fotoGrabar(@RequestParam("archivo")
CommonsMultipartFile archivo,
        @RequestParam("codigoUsuario") String codigoUsuario) {

     UsuarioEntity usuario = usuarioService.getUsuario(codigoUsuario);
```

```
        usuario.setFoto(archivo.getBytes());

        return new ModelAndView("usuarioLista", "lista",
usuarioService.getListaUsuarios());
    }
```

5. Modifica el configurador del Dispatcher para inscribir un ViewResolver adicional, indicándole al framework que podrá manejar también formularios con envío de archivos.

```
        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>

    <bean id="multipartResolver"
          class="org.springframework.web.multipart.commons.CommonsMultipartResolver"/>
```

6. Agrega las siguientes dependencias en Maven para las librerías de upload.

```
        <dependency>
            <groupId>javax.xml.bind</groupId>
            <artifactId>jaxb-api</artifactId>
            <version>2.3.0</version>
        </dependency>
        <dependency>
            <groupId>commons-fileupload</groupId>
            <artifactId>commons-fileupload</artifactId>
            <version>1.3.1</version>
        </dependency>
    </dependencies>
```

7. Ahora crea la página inicial de la subida de las fotos llamada "**fotoUsuario.jsp**".

```
<%@page import="java.util.Base64"%>
<%@page import="edu.cibertec.capitulo3.entity.UsuarioEntity"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Foto de usuario</title>
  </head>
  <body>
    <h1>Foto del Usuario</h1>
    <% if (request.getAttribute("usuario") == null
            || ((UsuarioEntity) request.getAttribute("usuario")).getFoto() == null) { %>
    <h2>Usuario aún sin foto</h2>
```

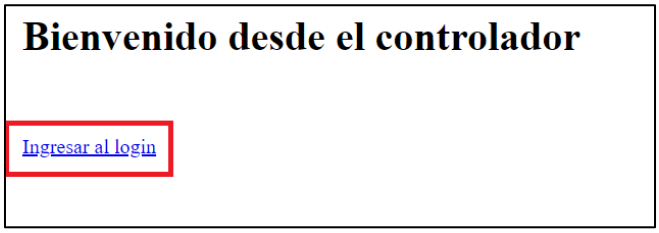
```
    <% } else {%>
        <img src="<%= "" +
              Base64.getEncoder().encodeToString(((UsuarioEntity)
                request.getAttribute("usuario")).getFoto())%>" />
    <% } %>
    <br />
    <form method="post" action="fotoGrabar.htm" enctype="multipart/form-data">
      <table>
        <tr>
          <td>Selecciona foto: </td>
          <td><input type="file" name="archivo"></td>
        </tr>
        <tr>
        </tr>
        <tr><td colspan="2" align="center">
              <input type="submit" value="Subir foto en jpg"></td>
        </tr>
      </table>
      <input type="hidden" name="codigoUsuario"
           value="<%= ((UsuarioEntity) request.getAttribute("usuario")).getUsuario()
%>">
    </form>
  </body>
</html>
```

8. Modifica la página de lista de usuarios para poner un enlace en cada registro y pueda subirse la foto.

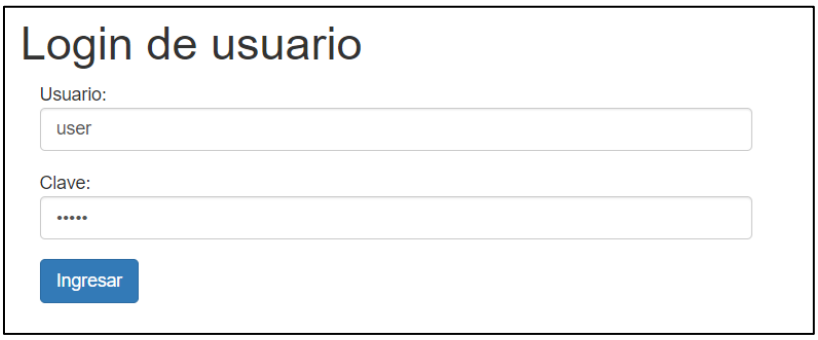
```
<thead>
    <tr class="info">
        <th>Usuario</th>
        <th>Clave</th>
        <th>Nombre completo</th>
        <th>Foto</th>
    </tr>
</thead>
<tbody>
    <% for (UsuarioEntity usuario : lista) {%>
    <tr>
        <td><%= usuario.getUsuario()%></td>
        <td><%= usuario.getClave()%></td>
        <td><%= usuario.getNombreCompleto()%></td>
        <td><a href="fotoMostrar.htm?codigoUsuario=<%= usuario.getUsuario()%>">foto</a></td>
    </tr>
    <% }%>
</tbody>
```

9. Ejecuta la aplicación y sigue los pasos.

   Selecciona el enlace de la primera página.



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



   Se muestra la página con el listado de los usuarios, presiona sobre uno de los enlaces de foto.

   La página muestra el mensaje que el usuario todavía no posee una foto asociada. Asimismo, presiona sobre el botón "**Seleccionar archivo**", escoge un archivo del tipo "**jpg**" y presiona sobre el botón "**Subir foto en jpg**".

La aplicación regresa a la lista de usuarios, vuelve a presionar sobre el enlace de "**fotos**" del usuario seleccionado anteriormente y visualiza la imagen cargada.

Como último paso, regresa a la lista de usuarios, presiona sobre otro enlace de foto (de otro usuario) y comprueba que muestra el mensaje que no posee foto asociada.

SESSION

1. En el controlador añade una "propiedad" que maneje el número de creaciones que se han dado en el sistema.

   Agrega la variable de sesión "**contador**" en la cabecera de la clase controladora y aumenta su valor cada vez que se crea un nuevo usuario de la siguiente manera.

```
@Controller
@SessionAttributes("contador")
public class UsuarioController {

    @Autowired
    private UsuarioService usuarioService;

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

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

        UsuarioEntity ue = usuarioService.validaLogin(usuarioValida);
```

```java
@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());
        mv.addObject("contador", 0);
    }
    return mv;
}
```

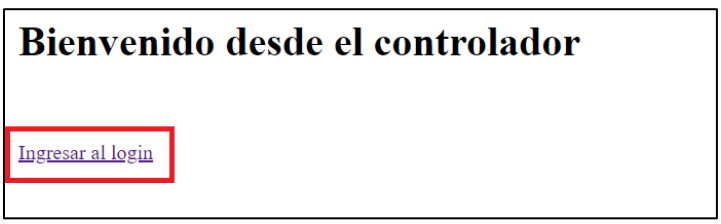
```java
@RequestMapping("usuarioGrabar")
public ModelAndView usuarioGrabar(@Valid @ModelAttribute("usuarioBean")
        UsuarioEntity usuarioValida, BindingResult result, ModelMap modelMap) {

    ModelAndView mv = null;
    if (result.hasErrors())
    {
        mv = new ModelAndView("usuarioDatos", "usuarioBean", usuarioValida);
    }
    else
    {
        usuarioService.insertaUsuario(usuarioValida);
        int contador = (int)modelMap.get("contador");
        contador++;

        mv = new ModelAndView("usuarioLista", "lista", usuarioService.getListaUsuarios());
        mv.addObject("contador", contador);
    }
    return mv;
}
```

2. En la página de lista de usuarios se muestra el contador en la parte inferior. Luego, añade la siguiente línea en el archivo "**usuarioLista.jsp**".

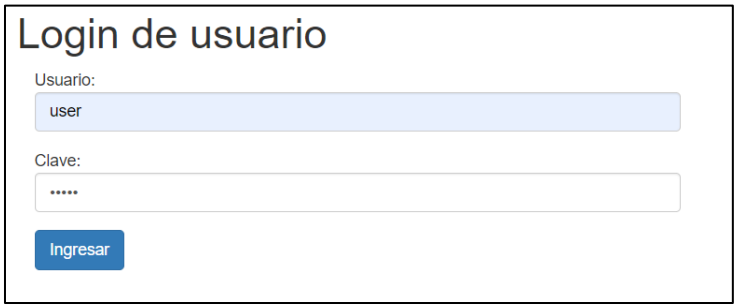
```jsp
                <% for (UsuarioEntity usuario : lista) {%>
                <tr>
                    <td><%= usuario.getUsuario()%></td>
                    <td><%= usuario.getClave()%></td>
                    <td><%= usuario.getNombreCompleto()%></td>
                    <td><a href="fotoMostrar.htm?codigoUsuario=<%= usuario.getUsuario()%>">foto</a>
                </tr>
                <% }%>
            </tbody>
        </table>
    </div>
    <a href="usuarioCrear.htm" class="btn btn-primary btn-lg active">Crear Usuario</a>
    <h3>Se han creado en esta sesión ${sessionScope.contador} usuarios! </h3>
</body>
```

3. Ejecuta la aplicación y sigue los siguientes pasos.

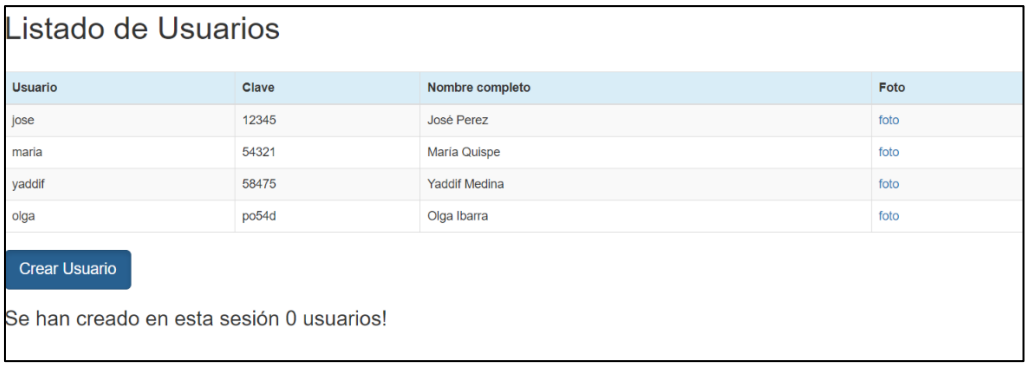
   Haz clic para ingresar al login.



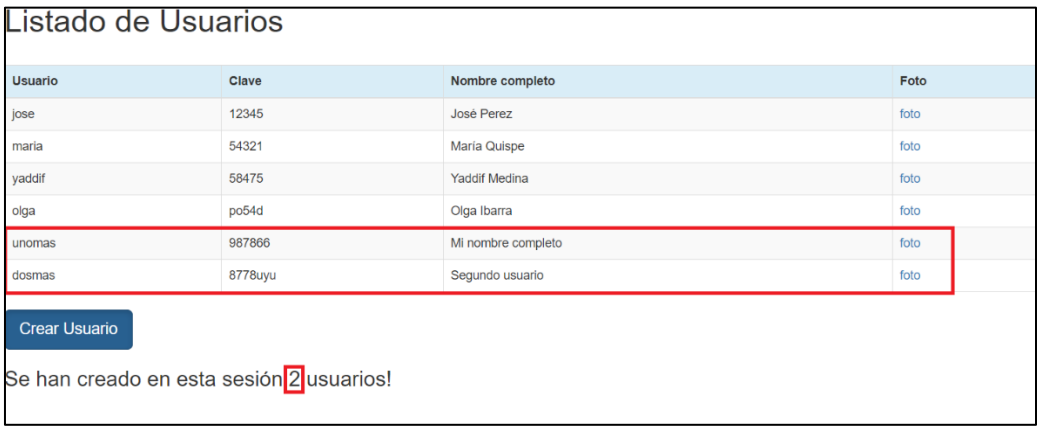
Ingresa el usuario y clave de manera correcta.



Se muestra la lista y el contador de usuario empezando en cero.



Añade varios usuarios y observa cómo el contador se acumula.



IV. EVALUACIÓN

Añade lo siguiente al ejercicio.

1. Mostrar en la columna de “foto” en la lista del usuario, la imagen en un tamaño miniatura.
2. Poder subir otro tipo de gráfico a la foto y no solo JPG.