



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

Objetivos de la sesión:

- Aprender a configurar **iconos (fuentes Awesome) en los botones.**
- Aprender a incorporar **imágenes estáticas** para poder mostrarlas en nuestro proyecto Spring.
- Entender la diferencia entre **imágenes estáticas y dinámicas.**
- Aprender a incorporar **imagenes dinámicas** para cambiar la foto del usuario en nuestra webapp.
- Ser capaces de **subir al servidor y descargar cualquier tipo de fichero** para poder incorporar documentación de un alumno.
- Crear **validadores de datos personalizados** para validar la imagen del usuario o la documentación del usuario a subir.



DWES
Home
Alumnos
Modulos
Errores

joseramon

Listado de alumnos:

Dni:
Ciclo:
Horario:

Dni	Nombre	Edad	Ciclo	Curso	Erasmus	Acciones
11111111A	Jose	21	DAM	1	<input type="checkbox"/>	<input type="button" value="Modificar"/> <input type="button" value="Borrar"/> <input type="button" value="Documentación"/>
22222222B	Pedro	32				
33333333C	Juan	23				

Imagen del usuario:



Puede actualizar la imagen seleccionando una ni

DWES
Home
Alumnos
Modulos
Errores

joseramon

Documentación del alumno:

Dni	Nombre	Ciclo	Curso
11111111A	Jose	DAM	1

Si desea añadir nueva documentación introduzca los datos:

Id:
Tipo:
☐ Certificado
☐ Justificante
☐ Solicitud

Comentario:

Seleccione el archivo: No se ha seleccionado ningún archivo.

Id	Tipo	Comentario	
1	Certificado (pdf)	Certificado en PDF	<input type="button" value="Descargar"/>
2	Justificante (png)	Captura de pantalla del justificante	<input type="button" value="Descargar"/>



Vamos a empezar viendo como podemos incorporar iconos e imágenes en nuestra aplicación web.

Empezaremos poniendo iconos a nuestro botones gracias a la librería de fuentes 'Awesome':



Listado de alumnos:

Dni: Ciclo: Horario:

Dni	Nombre	Edad	Ciclo	Curso	Erasmus	Acciones
11111111A	Jose	21	DAM	1	<input type="checkbox"/>	<input type="button" value="Modificar"/> <input type="button" value="Borrar"/> <input type="button" value="Documentación"/>
22222222B	Pedro	32	DAW	2	<input type="checkbox"/>	<input type="button" value="Modificar"/> <input type="button" value="Borrar"/> <input type="button" value="Documentación"/>
33333333C	Juan	23	ASIR	1	<input type="checkbox"/>	<input type="button" value="Modificar"/> <input type="button" value="Borrar"/> <input type="button" value="Documentación"/>



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

• Fuentes Awesome: Iconos:

Para poder **incorporar esta librería** debemos añadir la dependencia al pom.xml:

```
joseramon_primer_app_spring_mvc/pom.xml
24<dependency>
25    <groupId>org.webjars</groupId>
26    <artifactId>bootstrap</artifactId>
27    <version>4.5.2</version>
28</dependency>
29<dependency>
30    <groupId>org.hibernate</groupId>
31    <artifactId>hibernate-validator</artifactId>
32    <version>6.1.6.Final</version>
33</dependency>
34<dependency>
35    <groupId>org.webjars</groupId>
36    <artifactId>font-awesome</artifactId>
37    <version>5.15.1</version>
38</dependency>
39</dependencies>
40<build>
41    <pluginManagement>
42        <plugins>
43            <plugin>
44                <groupId>org.apache.maven.plugins</groupId>
45                <artifactId>maven-compiler-plugin</artifactId>
```

Vale la pena poner exactamente la misma versión que la imagen (5.15.1) para evitar posibles problemas ocasionados por cambiar de una versión a otra.

Y debemos añadir la referencia a la librería en header.jspf

```
header.jspf
1 <%@taglib uri="http://www.springframework.org/tags/form" prefix="mvc"%>
2 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <link href="webjars/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
7 <link href="webjars/font-awesome/5.15.1/css/all.min.css" rel="stylesheet">
8 <title>${pagina.getTitulo()}</title>
```




UD 2: Modelo Vista Controlador

10.- Ficheros








Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

• Fuentes Awesome: Iconos:

Ahora descomprimimos el fichero DWES_UD2_08_Iconos_Imagenes.zip y **actualizamos los 2 ficheros jsp** con los del fichero comprimido. Como resultado ya podemos ver la siguiente página :

De momento no copiar el fichero jpg a ningún sitio!!

 DWES  Home  Alumnos  Modulos  Errores

Introduzca su usuario:

Introduzca su contraseña:

DWES: Desarrollo Web en Entorno Servidor - profesor: josemon.profesor@gmail.com

La parte derecha hace una cosa rara , pero hasta que no veamos como hacer referencia a imagenes no podemos arreglarlo.



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

• Fuentes Awesome: Iconos:

No vamos a profundizar en la explicación de la parte visual. Se deja al alumno investigar como funcionan las fuentes awesome consultando la página menuSuperior.jspf y el enlace : <https://fontawesome.com/v4.7.0/examples/>. Ojo: Al ser una versión diferente puede que algunas cosas no funcionen exactamente igual.

```
menuSuperior.jspf
13 <li class="nav-item ${pagina.getStrBootstrapActiva("login")}">
14 <a class="nav-link" href="login"><i class="fas fa-home"></i>&nbsp;Home
15 </a>
16 </li>
17 <li class="nav-item ${pagina.getStrBootstrapActiva("list-alumno")}">
18 <a class="nav-link" href="list-alumno">
19 <i class="fas fa-user-friends"></i>&nbsp;Alumnos
20 </a>
21 </li>
22 <li class="nav-item ${pagina.getStrBootstrapActiva("list-modulo")}">
23 <a class="nav-link" href="list-modulo">
24 <i class="fas fa-book"></i>&nbsp;Modulos
25 </a>
26 </li>
27 <li class="nav-item ${pagina.getStrBootstrapActiva("list-logerror")}">
28 <a class="nav-link" href="list-logerror">
29 <i class="fas fa-skull-crossbones"></i>&nbsp;Errores</a>
30 </li>
31 </div>
32 <div class="nav-item dropdown">
33 <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-t
34 
37 <div class="dropdown-menu" aria-labelledby="navbarDropdown">
38 <a class="dropdown-item" href="#">
39 <i class="fas fa-pen"></i>&nbsp; Editar
40 </a>
41 <a class="dropdown-item" href="#">
42 <i class="fas fa-user-circle"></i>&nbsp; Foto
43 </a>
44 <div class="dropdown-divider"></div>
45 <a class="dropdown-item text-danger" href="login">
46 <i class="fas fa-door-open"></i>&nbsp; Logout
47 </a>
48 </div>
```



UD 2: Modelo Vista Controlador

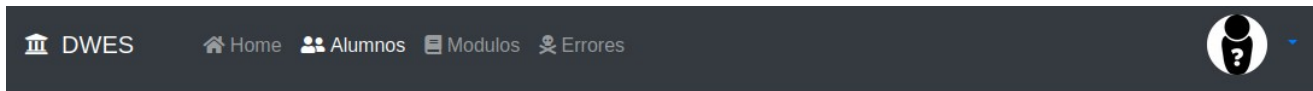
10.- Ficheros



Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

· Imágenes:

Nuestra intención es mostrar la imagen “usuarioSinFoto.jpg” cuando el usuario logeado no tenga foto:



Listado de alumnos:

Dni: Ciclo: Horario:

Dni	Nombre	Edad	Ciclo	Curso	Erasmus	Acciones
11111111A	Jose	21	DAM	1	<input type="checkbox"/>	<input type="button" value="Modificar"/> <input type="button" value="Borrar"/> <input type="button" value="Documentación"/>
22222222B	Pedro	32	DAW	2	<input type="checkbox"/>	<input type="button" value="Modificar"/> <input type="button" value="Borrar"/> <input type="button" value="Documentación"/>
33333333C	Juan	23	ASIR	1	<input type="checkbox"/>	<input type="button" value="Modificar"/> <input type="button" value="Borrar"/> <input type="button" value="Documentación"/>

```
<div class="nav-item dropdown">  
  <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown">  
      
    &nbsp;{{loginNickName}}  
  <div class="dropdown-menu">  
    <div class="dropdown-item">  
      <a href="#">Inicio</a>  
    </div>  
  </div>  
</div>
```

DWES: Desarrollo Web en Entorno Servidor - profesor: joseramon.profesor@gmail.com



¿Como añadimos una imagen en nuestro proyecto?



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

· Imágenes:

Para poder hacer referencia a una imagen en nuestro proyecto deberemos de **añadir la carpeta** donde encontraremos la lista de recursos estáticos (imagenes) en el fichero **alumno-servlet.xml**:

```
alumno-servlet.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:context="http://www.springframework.org/schema/context"
4       xmlns:mvc="http://www.springframework.org/schema/mvc"
5       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6       xsi:schemaLocation="http://www.springframework.org/schema/beans http://
7       http://www.springframework.org/schema/mvc http://www.springframework.o
8       http://www.springframework.org/schema/context http://www.springframewo
9       <context:component-scan base-package="org.profesor.joseramon" />
10      <mvc:annotation-driven />
11 <bean
12     class="org.springframework.web.servlet.view.InternalResourceViewResolv
13     <property name="prefix">
14         <value>/WEB-INF/views/</value>
15     </property>
16     <property name="suffix">
17         <value>.jsp</value>
18     </property>
19 </bean>
20 <mvc:resources mapping="/webjars/**" location="/webjars/" />
21 <mvc:resources mapping="/imagenes/**" location="/resources/imagenes/" />
22 </beans>
```




UD 2: Modelo Vista Controlador

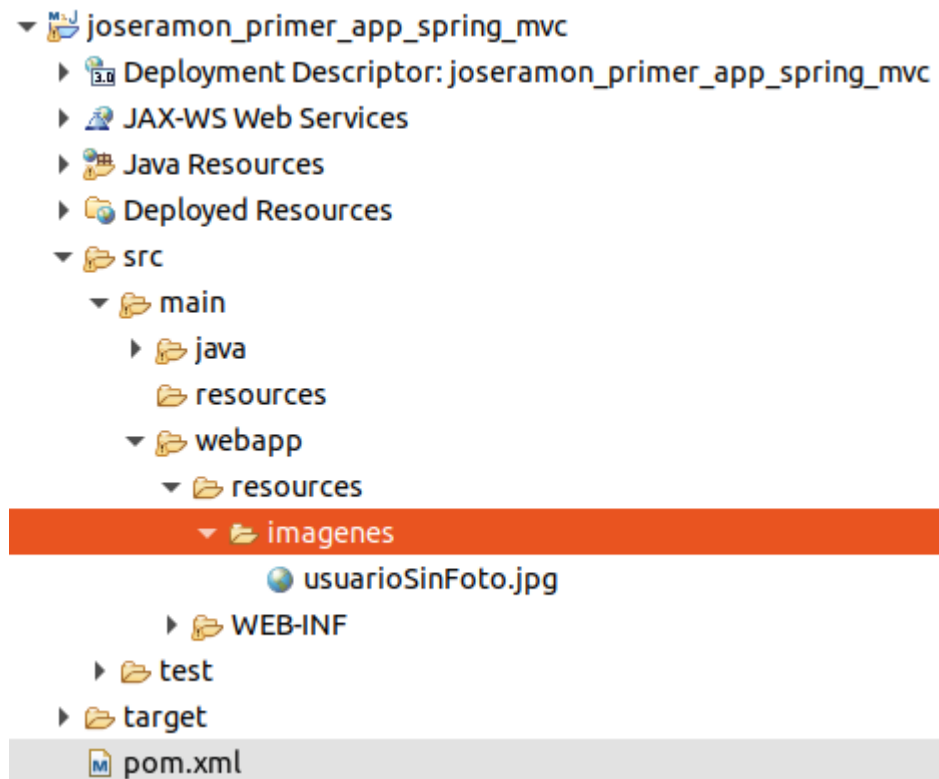
10.- Ficheros



Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

· Imagenes:

Ahora debemos crear la carpeta “src\main\webapp\resources\imagenes” y copiar el fichero jpg del fichero comprimido de esta práctica:





UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

Nombre del usuario:

Solo nos queda poder indicar el nombre del usuario en las páginas. Para ello deberemos de **añadir "loginName" y "loginNickName"** a la lista de variables de sesión para que desde **menuSuperior.jspf** podamos leer los valores:

También podemos leer los datos del "usuario", si lo prefieres...

DWES Home Alumnos Modulos Errores joseramon

Listado de alumnos:

Dni:
Ninguno

Ciclo:
Ninguno

Horario:
Ninguno

Filtrar

Dni	Nombre	Edad	Ciclo	Curso	Erasmus	Acciones
11111111A	Jose	21	DAM	1	<input type="checkbox"/>	<div>ModificarBorrarDocumentación</div>
22222222B	Pedro	32	DAW	2	<input type="checkbox"/>	<div>ModificarBorrarDocumentación</div>
33333333C	Juan	23	ASIR	1	<input type="checkbox"/>	<div>ModificarBorrarDocumentación</div>

Añadir alumno

```
<div class="nav-item dropdown">  
  <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown  
    
```



UD 2: Modelo Vista Controlador

10.- Ficheros




Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

Ahora ya sabemos como añadir imagenes estáticas. Ahora veremos como podemos incorporar **imágenes dinámicas** en nuestra aplicación web:

DWES

Home Alumnos Modulos Errores

 joseramon

Listado de alumnos:

Dni: Ninguno

Ciclo: Ninguno

Horario: Ninguno

Filtrar

Dni	Nombre	Edad	Ciclo	Curso	Erasmus	Acciones
11111111A	Jose	21	DAM	1	<input type="checkbox"/>	<div>Modificar</div> <div>Borrar</div> <div>Documentación</div>
22222222B	Pedro	32	DAW	2	<input type="checkbox"/>	<div>Modificar</div> <div>Borrar</div> <div>Documentación</div>
33333333C	Juan	23	ASIR	1	<input type="checkbox"/>	<div>Modificar</div> <div>Borrar</div> <div>Documentación</div>

Añadir alumno

DWES: Desarrollo Web en Entorno Servidor - profesor: joseramon.profesor@gmail.com



¿Que diferencia hay entre imágenes estáticas y dinámicas?



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

Las **imágenes estáticas** son imágenes que no cambian y se encuentran dentro de nuestra aplicación en una carpeta y forman parte del fichero comprimido .WAR que se ejecuta en Tomcat.

Las **imágenes dinámicas** son aquellas que son independiente de nuestra aplicación web, no están dentro del fichero .WAR sino que son proporcionadas por una Base de Datos o actualmente en nuestro caso por el sistema de ficheros del sistema operativo sobre el que se esta ejecutando nuestra aplicación web.

A continuación vamo a **personalizar la imagen del usuario** a mostrar cuando nos logeamos. Para ello realizaremos los siguientes **pasos**:

- 1º Añadir las dependencias para poder trabajar con ficheros
- 2º Permitir que la aplicación sirva imágenes de una carpeta del SO
- 3º Crear la vista, controlador y servicio para modificar la imagen de usuario
- 4º Crear un validador personalizado de imágenes de usuarios (anotación)

UD 2: Modelo Vista Controlador

10.- Ficheros

Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com



1º Añadir las dependencias para poder trabajar con ficheros:

Buscamos en Maven las últimas versiones estables de “**commons-fileupload**” y de “**commons-io**” y **añadimos las dependencias** a nuestro proyecto:

```
joseramon_primer_app_spring_mvc/pom.xml
33     </dependency>
34     <dependency>
35         <groupId>org.webjars</groupId>
36         <artifactId>font-awesome</artifactId>
37         <version>5.15.1</version>
38     </dependency>
39     <!-- Apache Commons FileUpload -->
40     <dependency>
41         <groupId>commons-fileupload</groupId>
42         <artifactId>commons-fileupload</artifactId>
43         <version>1.4</version>
44     </dependency>
45     <!-- Apache Commons IO -->
46     <dependency>
47         <groupId>commons-io</groupId>
48         <artifactId>commons-io</artifactId>
49         <version>2.8.0</version>
50     </dependency>
51 </dependencies>
52 <build>
53     <pluginManagement>
54         <plugins>
55             <plugin>
56                 <groupId>org.apache.maven.plugins</groupId>
57                 <artifactId>maven-compiler-plugin</artifactId>
```



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

... continuación **1º Añadir las dependencias para poder trabajar con ficheros:**

Para que podamos cargar un fichero desde el formulario y enviarlo al Controller deberemos de indicar a Spring quien se encarga de atender los ficheros multipart:

alumno-servlet.xml

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:context="http://www.springframework.org/schema/context"
4       xmlns:mvc="http://www.springframework.org/schema/mvc"
5       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
6       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.spr
7       http://www.springframework.org/schema/mvc http://www.springframework.org/schem
8       http://www.springframework.org/schema/context http://www.springframework.org/sc
9       <context:component-scan base-package="org.profesor.josemon" />
10      <mvc:annotation-driven />
11 <bean
12     class="org.springframework.web.servlet.view.InternalResourceViewResolver">
13     <property name="prefix">
14         <value>/WEB-INF/views/</value>
15     </property>
16     <property name="suffix">
17         <value>.jsp</value>
18     </property>
19 </bean>
20 <bean id="multipartResolver"
21     class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
22 </bean>
23 <mvc:resources mapping="/webjars/**" location="/webjars/" />
24 <mvc:resources mapping="/imagenes/**" location="/resources/imagenes/" />
25 </beans>
```



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com

2º Permitir que la aplicación sirva imágenes de una carpeta del SO:

El objetivo es cambiar la imagen estática “usuarioSinFoto.jpg” por una almacenada en una carpeta del sistema operativo. Dicha imagen se podrá configurar en el usuario y se visualizará llamando a la **ruta “imagenUsuario/nickname”**.

La idea es que leyendo nickname podamos buscar la imagen de ese usuario. Para ello **cambiamos la vista menuSuperior.jspf para mostrar una ruta dinámica:**

```
menuSuperior.jspf
26     </li>
27     <li class="nav-item ${pagina.getStrBootstrapActiva("list-logerror")}">
28     <a class="nav-link" href="list-logerror">
29         <i class="fas fa-skull-crossbones"></i>&nbsp;&nbsp;&Errores</a>
30     </li>
31 </div>
32 <div class="nav-item dropdown">
33     <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true"
34         
35         &nbsp;&nbsp;&${loginNickName}
36     </a>
37     <div class="dropdown-menu" aria-labelledby="navbarDropdown">
38     <a class="dropdown-item" href="update-usuario?nickName=${loginNickName}">
39         <i class="fas fa-pen"></i>&nbsp;&nbsp;& Editar
40     </a>
41     <a class="dropdown-item" href="update-imagenUsuario?nickName=${loginNickName}">
42         <i class="fas fa-user-circle"></i>&nbsp;&nbsp;& Foto
43     </a>
44     <div class="dropdown-divider"></div>
45     <a class="dropdown-item text-danger" href="login">
46         <i class="fas fa-door-open"></i>&nbsp;&nbsp;& Logout
47     </a>
48 </div>
49 </div>
50 </nav>
```




2º Permitir que la aplicación sirva imágenes de una carpeta del SO:

El nombre del fichero con la foto deberá almacenarse en el **bean “Usuario”** por lo que debemos **añadir nuevos campos e implementar métodos nuevos relacionados con los getters, los setters y la interfaz Modificable**:

- **“nombreFicheroConImagen”**: Se permitirá subir un fichero con extensión png, jpg o gif. Cuando subamos al servidor una imagen se guardará siempre en el mismo directorio y le cambiaremos el nombre al fichero original para que tenga el nombre “nickname.extension” donde extensión será un png, jpg o un gif.

- **“ts” y “user”**: Permitirá indicar quien ha modificado el usuario y cuando.

```

Usuario.java
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.model;
2
3 import java.io.Serializable;
11
12 public class Usuario implements Serializable, Modificable<Usuario>{
13     private static final long serialVersionUID = 1L;
14     @Size(min=5,message="El usuario debe de tener un tamaño mínimo de 5 caracteres")
15     private String nickname;
16     private String nombre;
17     @Pattern(regexp = "(?=.*\\d)(?=.*[\\u0021-\\u002b\\u003c-\\u0040])(?=.*[A-Z])(?=.*[a-z])\\S{8,16}$",
18     private String password;
19     private String nombreFicheroConImagen;//contendrá 'nickname.ext' donde 'ext' será JPG,GIF o PNG.
20     private Date ts; //almacena la fecha de la última modificación
21     private String user; //almacena la persona que ha realizado la última modificación
22

```




UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

2º Permitir que la aplicación sirva imágenes de una carpeta del SO:

Crear el servicio **"FileService.java"** que nos devolverá el fichero a mostrar para atender la peticiones de la vista **"imagenUsuario/nickname"**.

Leer los comentarios (en verde) del código para entender los siguientes pasos:

```
@Service //Este trozo de código puede copiarse desde el PDF proporcionado al alumno!!
public class FileService {
    //En los FICHEROS ESTÁTICOS, los datos se almacenan en una carpeta dentro del fichero
    //comprimido .WAR que se ha cargado en Tomcat.
    //En los FICHEROS DINÁMICOS los ficheros se almacenan fuera del fichero comprimido .WAR,
    // esto significa que se guardará en una CARPETA DEL SERVIDOR WEB

    /** CONSTANTES CON DATOS PARA SABER LA CARPETA PRINCIPAL DE LA APP DONDE GUARDAR/LEER LOS FICHEROS */
    //Podemos almacenar los ficheros dinámicos donde queramos, pero si ponemos una
    //carpeta fija, dicha ruta tiene codificación Windows o Linux, por lo que nuestra app
    //solo podríamos ejecutarla en ese sistema operativo.
    //La carpeta $HOME del usuario se puede extraer gracias a la clase System
    //independientemente de que estemos en Windows, Linux o Mac.
    private static final String USER_HOME_TOMCAT= System.getProperty("user.home");
    //En linux el separador de carpetas es "/", en windows "\", para hacer nuestra APP
    //independiente del sistema operativo donde se cargue utilizamos el File.separator.
    private static final String SEPARATOR= File.separator;
    //Carpeta donde se almacenaran todos los ficheros dinámicos de la app
    private static final String CARPETA_FICHEROS_DINAMICOS_WEBAPP=USER_HOME_TOMCAT+SEPARATOR+"AlumnosWebApp_DynamicFiles"
                                                                    +SEPARATOR;

    /** CONSTANTES CON EL NOMBRE DE LAS CARPETAS DONDE GUARDAR LOS FICHEROS: */
    // Carpeta con las imagenes de los usuarios
    private static final String CARPETA_IMAGENES_USUARIOS=CARPETA_FICHEROS_DINAMICOS_WEBAPP+"ImagenesUsuarios";
    // Carpeta con las imagenes de los usuarios
    private static final String CARPETA_DOCUMENTACION_ALUMNOS=CARPETA_FICHEROS_DINAMICOS_WEBAPP+"DocumentacionAlumnos";

    public FileSystemResource getImagenUsuario(String fichero) {
        return new FileSystemResource(new File(CARPETA_IMAGENES_USUARIOS, fichero));
    }
}
```



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

2º Permitir que la aplicación sirva imágenes de una carpeta del SO:

Crear el controlador “UsuarioController.java” que nos servirá para atender la peticiones de la vista “imagenUsuario/nickname”.

Para hacer referencia a dicha imagen utilizaremos /imagenUsuario/{nickname}, aparenciando una nueva notación de Spring **@PathVariable**.

//Este trozo de código puede copiarse desde el PDF proporcionado al alumno!!

```
@RequestMapping(value = "/imagenUsuario/{nickname}" , method = RequestMethod.GET)
public ResponseEntity<FileSystemResource> getFile(@PathVariable("nickname") String nickname) {
    try {
        String nombreFicheroConImagen="Desconocido.jpg";//valor por defecto
        if (!"Desconocido".contentEquals(nickname)) { //Si no es el valor por defecto
            //La imagen siempre se almacenará como 'nickname.extensionImagen',
            //con la posibilidad de que extensionImagen sea JPG, PNG o GIF.
            // Para saber el formato consultaremos 'Usuario' para conseguir el nombre del fichero
            Usuario usuarioAMostrarFoto=loginService.encontrarUsuarioPorNickName(nickname);
            nombreFicheroConImagen=usuarioAMostrarFoto.getNombreFicheroConImagen();
        }
        //El servicio nos devolverá la imagen y nos abstraee de saber donde esta guardada realmente
        FileSystemResource resource = fileService.getImagenUsuario(nombreFicheroConImagen);
        if (!resource.exists()) {
            throw new Exception("La imagen no existe");
        }
        ResponseEntity<FileSystemResource> responseEntity = new ResponseEntity<>(resource, HttpStatus.OK);
        return responseEntity;
    } catch (Exception e) {//Ante cualquier error
        //Devolver error 404-recurso no encontrado
        throw new ResponseStatusException(HttpStatus.NOT_FOUND);
    }
}
```



UD 2: Modelo Vista Controlador

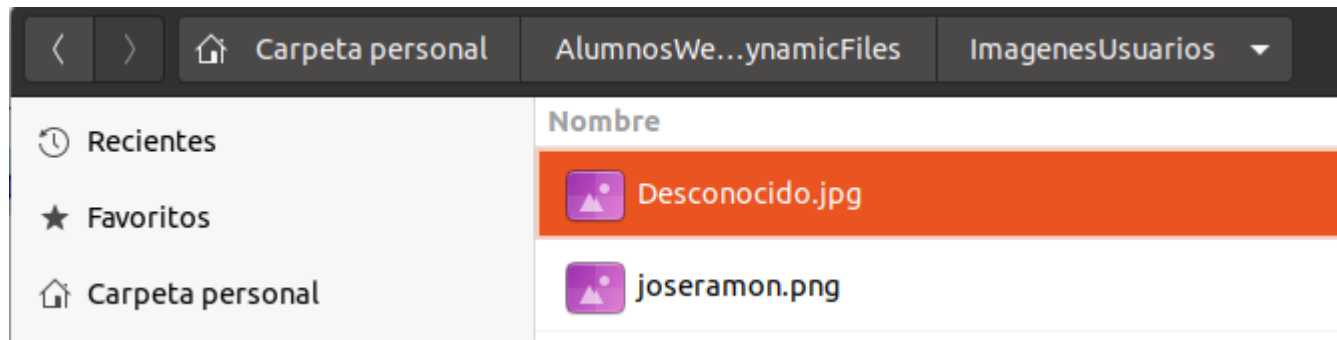
10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

2º Permitir que la aplicación sirva imágenes de una carpeta del SO:

Por último, para que no falle la aplicación deberá de existir la carpeta donde se almacenaran las imágenes en el sistema operativo del servidor web donde esté ejecutandose la aplicación . **Crear dicha carpeta y copiar usuarioSinFoto.jpg como “Desconocido.jpg”:**



```
/** CONSTANTES CON DATOS PARA SABER LA CARPETA PRINCIPAL DE LA APP DONDE GUARDAR/LEER LOS FICHEROS */
//Podemos almacenar los ficheros dinámicos donde queramos, pero si ponemos una
//carpeta fija, dicha ruta tiene codificación Windows o Linux, por lo que nuestra app
//solo podríamos ejecutarla en ese sistema operativo.
//La carpeta $HOME del usuario se puede extraer gracias a la clase System
//independientemente de que estemos en Windows, Linux o Mac.
private static final String USER_HOME_TOMCAT= System.getProperty("user.home");

//En linux el separador de carpetas es "/", en windows "\", para hacer nuestra APP
//independiente del sistema operativo donde se cargue utilizamos el File.separator.
private static final String SEPARATOR= File.separator;







//Carpeta donde se almacenaran todos los ficheros dinámicos de la app
private static final String CARPETA_FICHEROS_DINAMICOS_WEBAPP=USER_HOME_TOMCAT+SEPARATOR+"AlumnosWebApp_DynamicFiles"+SEPARATOR;

/** CONSTANTES CON EL NOMBRE DE LAS CARPETAS DONDE GUARDAR LOS FICHEROS: */
// Carpeta con las imagenes de los usuarios
private static final String CARPETA_IMAGENES_USUARIOS=CARPETA_FICHEROS_DINAMICOS_WEBAPP+"ImagenesUsuarios";
```




2º Permitir que la aplicación sirva imágenes de una carpeta del SO:







Ahora ya podremos probar la aplicación para ver si se muestra la imagen cuando no se está logeado y se muestra la misma imagen cuando te logeas (a falta de actualizar la imagen):

 DWES  Home  Alumnos  Modulos  Errores  Desconocido ▾

Introduzca su usuario:

Introduzca su contraseña:

 Login

 DWES  Home  Alumnos  Modulos  Errores  joseramon ▾

Bienvenid@ JoseRa



UD 2: Modelo Vista Controlador

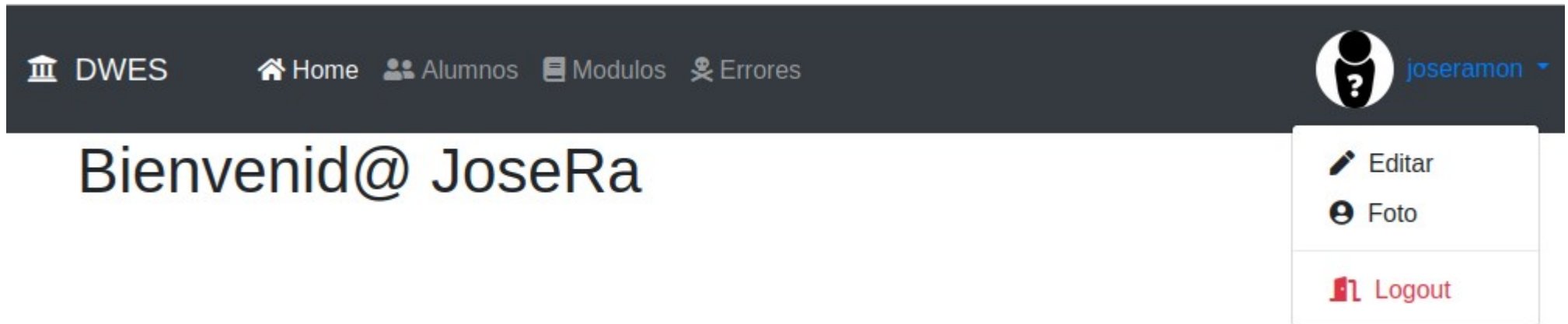
10.- Ficheros

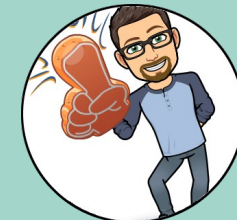
Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com



3º Crear la vista, controlador y servicio para modificar la imagen de usuario:

Nuestra intención es que cuando nos logeemos podamos modificar la foto pulsando sobre el submenu “Foto”:





3º Crear la vista, controlador y servicio para modificar la imagen de usuario:

Crear una vista “update-imagenUsuario.jsp” que muestre la imagen actual y permita cambiarla:

Deberemos de enviarle un bean nuevo “imagenUsuario” que contenga “nickname” (String con el nickname del usuario) y un campo “imagen” que es de tipo “MultipartFile” (que utilizaremos para seleccionar el fichero a subir al servidor):

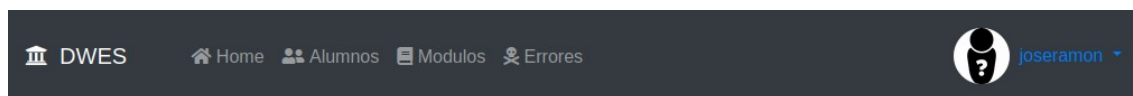


Imagen del usuario:



Puede actualizar la imagen seleccionando una nueva imagen: No se ha seleccionado ningún archivo.

```
update-imagenUsuario.jsp
1 <%@taglib uri="http://www.springframework.org/tags/form" prefix="mvc"%>
2 <%@ include file="../../jspf/header.jspf"%>
3 <%@ include file="../../jspf/menuSuperior.jspf"%>
4
5 <div class="container">
6   <h1>Imagen del usuario:</h1>
7   <p>
8     <font color="red">${errores}</font>
9   </p>
10  
11  <br>
12  <mvc:form method="post" action="guardar-imagen-usuario" enctype="multipart/form-data" modelAttribute="imagenUsuario">
13    <mvc:hidden path="nickname" />
14    <mvc:label path="imagen">Puede actualizar la imagen seleccionando una nueva imagen:</mvc:label>
15    <mvc:input path="imagen" type="file" /><form:errors path="imagen"/><br>
16    <button type="submit" class="btn btn-success"><i class="fas fa-user-edit"></i>&nbsp;Actualizar</button>
17  </mvc:form>
18 </div>
19
20 <%@ include file="../../jspf/footer.jspf"%>
21
```



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

3º Crear la vista, controlador y servicio para modificar la imagen de usuario:

En FileService deberemos de **crear un método “guardarFichero”** para guardar cualquier tipo de fichero en cualquiera de las rutas configuradas (de momento solo una , pero habrán más en próximas prácticas) :

//Este trozo de código puede copiarse desde el PDF proporcionado al alumno!!

```
/** METODO PRINCIPAL AL QUE LLAMARAN TODOS LOS MÉTODOS QUE GUARDEN FICHEROS */  
// Guardar el 'fichero' del formulario en la 'ruta'.  
// Si falla devuelve un String con el error, en caso contrario nulo.  
// 'ruta' ya incluye el nombre final del fichero.  
// Ejemplo:  
// ruta='/home/joseramon/AlumnosWebApp_DynamicFiles/ImagenesUsuarios/imagenJoseRamon.jpg'  
private String guardarFichero(String ruta, MultipartFile fichero) {  
    try {  
        //Obtener datos del fichero  
        byte[] fileBytes= fichero.getBytes();  
        //Obtener ruta  
        Path path=Paths.get(ruta);  
        //Guardar fichero  
        //El fichero no tiene porque existir en la ruta, pero  
        //si la carpeta destino no existe se producirá una excepción  
        Files.write(path, fileBytes);  
    } catch (NoSuchFileException e) {  
        return "No existe la carpeta para poder guardar '"+e.getMessage()+"';"  
    } catch (IOException e) {  
        //Si hay error devolver mensaje de error  
        return e.getMessage();  
    }  
    //Si no hay error devolver null  
    return null;  
}
```



3º Crear la vista, controlador y servicio para modificar la imagen de usuario:

Adicionalmente también deberemos de crear el método que permitirá guardar la imagen del usuario y otro que nos proporciona el nombre del nuevo fichero:

//Este trozo de código puede copiarse desde el PDF proporcionado al alumno!!

```
/** Guardar imagenes de los usuarios */
//Guardar fichero en la carpeta de las imagenes de usuario con un nombre determinado
//Si falla devuelve una lista con los errores detectados
public ArrayList<String> guardaImagenUsuario(MultipartFile fichero,String nickName) {
    String nombreFichero= getNombreImagenUsuario(fichero,nickName);

    //Comprobaciones de errores
    if (!ValidadorImagenes.imagenValida(fichero)) {
        return ValidadorImagenes.mensajesErrorImagen(fichero);
    }

    //Guardar fichero
    String errorAlGuardar=guardarFichero(CARPETA_IMAGENES_USUARIOS+SEPARATOR+nombreFichero,fichero);
    if (errorAlGuardar==null)
        return new ArrayList<String>();
    else
        return new ArrayList<String>(List.of(errorAlGuardar));
}

//Consulta el tipo de extensión del fichero y devuelve un String "nickname.ext"
public String getNombreImagenUsuario(MultipartFile fichero,String nickName) {
    String extension= ValidadorImagenes.getExtension(fichero);
    String nombreFichero= nickName+"."+ extension;
    return nombreFichero;
}
```

Aparece en escena una clase nueva “ValidadorImagenes” que deberemos de crear en el subpaquete “validaciones”.



UD 2: Modelo Vista Controlador

10.- Ficheros

Desarrollo Web en Entorno Servidor - Jose Ramon.profesor@gmail.com



3º Crear la vista, controlador y servicio para modificar la imagen de usuario:

En el validador de imágenes de momento no hace falta hacer el “implements”.

Copia el código entendiendolo y crea tu solo los métodos en negrita:

- mensajesErrorImagen
- **tipoDeImagenValido:**
- imagenValida
- **getExtension :**

Para saber el tipo del fichero puedes utilizar `getContentType()` Por ejemplo , si el `contentType` es “image/jpg” devolverás “jpg”.

```
ValidadorImagenes.java
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.validaciones;
2
3 import org.springframework.web.multipart.MultipartFile;
4
5
6
7
8
9
10
11 public class ValidadorImagenes implements ConstraintValidator<ImagenValida, MultipartFile> {
12     public static final List<String> tiposDeImagenes=Arrays.asList("image/png", "image/jpg", "image/jpeg", "image/gif");
13     public static final long MAX_BYTES= 524288;//512 KB de tamaño máximo
14     public void initialize(ImageValida constraintAnnotation) {}
15
16
17
18
19
20
21     public boolean isValid(MultipartFile multipartFile, ConstraintValidatorContext context) {}
22
23
24
25     public static ArrayList<String> mensajesErrorImagen(MultipartFile fichero) {
26
27         ArrayList<String> errores=new ArrayList<String>();
28
29         //Fichero no vacio
30         if (fichero.isEmpty()) {
31             errores.add("La imagen no puede estar vacia");
32         }
33
34         //Validar tipo de fichero
35         String contentType = fichero.getContentType();
36         if (!tipoDeImagenValido(contentType)) {
37             errores.add("Solo se permiten imagenes PNG, JPG o GIF.");
38         }
39
40         //Comprobar tamaño máximo
41         if (fichero.getSize()>MAX_BYTES) {
42             errores.add("Tamaño máximo de la imagen excedido (" +MAX_BYTES+" bytes)");
43         }
44
45         return errores;
46     }
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2
```



UD 2: Modelo Vista Controlador

10.- Ficheros

Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com



3º Crear la vista, controlador y servicio para modificar la imagen de usuario:

En UsuarioController deberemos de crear 2 métodos:

1º Método para visualizar la página con la imagen del usuario:

```
@RequestMapping(value = "/update-imagenUsuario", method = RequestMethod.GET)
public String updateImagenUsuario(ModelMap model, @RequestParam String nickName) {
```

Como dicho método será el encargado de mostrar el jsp donde visualizaremos la foto y podremos seleccionar otra desde el formulario (update-imagenUsuario.jsp) para visualizar la página con la imagen del usuario solo deberemos de pasar en el modelo un bean de tipo "ImagenUsuario" con el nickname relleno y hacer el return hacia la página jsp.



UD 2: Modelo Vista Controlador

10.- Ficheros

Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com



3º Crear la vista, controlador y servicio para modificar la imagen de usuario:

2º Método para recibir el fichero seleccionado:

```
@RequestMapping(value="/guardar-imagen-usuario",method = RequestMethod.POST)  
public String guardarImagenUsuario(ModelMap model,  
    @Valid ImagenUsuario imagenUsuario, BindingResult validacion) {
```

A continuación se muestra el código, pero antes debemos entender que vamos a hacer:

- Validar si hay errores (de momento solo configurar en el bean que el nickname sea de tamaño mínimo 5 y el fichero no sea nulo)
- Comprobar si el usuario esta logeado (lo tenemos como variable de sesión)
- Llamar a guardarmImagenUsuario de FileService.
- Si no hay errores se actualizará la imagen del usuario (atributo “nombreFicheroConImagen”) utilizando el método “getNombreImagenUsuario” del servicio “FileService” y se procederá a guardar los cambios con el método “modificaUsuario” de “LoginService”.

En caso de que exista cualquier error mostrará la lista de errores en la misma pantalla que estaba (update-imagenUsuario.jsp).



3º Crear la vista, controlador y servicio para modificar la imagen de usuario:

2º Método para recibir el fichero seleccionado:

```
@RequestMapping(value="/guardar-imagen-usuario",method = RequestMethod.POST)
public String guardarImagenUsuario(ModelMap model,
    @Valid ImagenUsuario imagenUsuario, BindingResult validacion) {
    if (validacion.hasErrors()) {
        // Hay errores y debemos volver al formulario de modificación
        return "update-imagenUsuario";
    }
    // Si llega aquí no hay errores de validación

    String nickName=imagenUsuario.getNickname();
    MultipartFile fichero =imagenUsuario.getImagen();
    try {
        if (model.getAttribute("usuario") == null) {
            throw new Exception("Para realizar modificaciones debe estar logeado");
        }
        //Guardar la imagen y actualizar usuario
        //Si no se ha podido listaErroresAlGuardar no estará vacío
        ArrayList<String> listaErroresAlGuardar=fileService.guardaImagenUsuario(fichero, nickName);
        if(!listaErroresAlGuardar.isEmpty()) {
            //Rellenar los errores al intentar guardar para pasarselos a la excepcion
            String mensajeCompleto="";
            for(String mensaje:listaErroresAlGuardar) {
                mensajeCompleto+=mensaje+"<br>";
            }
            //lanzar excepción
            throw new Exception(mensajeCompleto);
        }
        //Si llega aquí se ha guardado correctamente la imagen y podemos actualizar el usuario
        Usuario quienModifica = (Usuario) model.getAttribute("usuario");
        Usuario usuarioAModificar=loginService.encontrarUsuarioPorNickName(nickName);
        usuarioAModificar.setNombreFicheroConImagen(fileService.getNombreImagenUsuario(fichero, nickName));
        loginService.modificaUsuario(usuarioAModificar, quienModifica.getNickname());
        // Para evitar pasar parámetros innecesarios
        model.clear();
        // Le pasamos el usuario actualizado
        model.addAttribute(loginService.encontrarUsuarioPorNickName(nickName));
        model.addAttribute("imagenUsuario", new ImagenUsuario(nickName));
        return "update-imagenUsuario";
    } catch (Exception e) {
        // Le pasamos el usuario actualizado
        model.addAttribute(loginService.encontrarUsuarioPorNickName(nickName));
        model.addAttribute("imagenUsuario", new ImagenUsuario(nickName));
        // Pasamos los errores
        model.addAttribute("errores", e.getMessage());
        // Hay errores y debemos volver al formulario de modificación
        return "update-imagenUsuario";
    }
}
```




4º Crear un validador personalizado de imágenes de usuarios (anotación)

Queremos poder añadir una anotación @ImagenValida al bean de la imagen del usuario que compruebe si la imagen cumple ciertas validaciones:

```
ImagenUsuario.java
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.model;
2
3 import java.io.Serializable;
4
5
6
7
8
9
10
11
12
13
14 public class ImagenUsuario implements Serializable{
15     @Size(min=5,message="El usuario debe de tener un tamaño mínimo de 5 caracteres")
16     private String nickname;
17     @NotNull
18     @ImagenValida
19     private MultipartFile imagen;
20 }
```

Para ello nos creamos una interface especial (@interface) llamada "ImagenValida.java":

//Este trozo de código puede copiarse desde el PDF proporcionado al alumno!!

```
package org.profesor.joseramon.joseramon_primer_app_spring_mvc.validaciones;
import javax.validation.Constraint;
import javax.validation.Payload;
import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;
// Información sobre como crear una anotación en Java:
// https://www.baeldung.com/java-custom-annotation
@Target(ElementType.FIELD)
@Retention(RetentionPolicy.RUNTIME)
@Constraint(validatedBy = {ValidadorImagenes.class})
public @interface ImagenValida {
    String message() default "Imagen incorrecta";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};
}
```



4º Crear un validador personalizado de imágenes de usuarios (anotación)

Y configuramos la clase “ValidadorImágenes” para que implemente dicha validación de restricciones con la herencia “ConstraintValidator” y la función “isValid”:

//Este trozo de código puede copiarse desde el PDF proporcionado al alumno!!

```
package org.profesor.joseramon.joseramon_primer_app_spring_mvc.validaciones;
import org.springframework.web.multipart.MultipartFile;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import javax.validation.ConstraintValidator;
import javax.validation.ConstraintValidatorContext;
public class ValidadorImágenes implements ConstraintValidator<ImagenValida, MultipartFile> {
    public static final List<String> tiposDeImagenes=Arrays.asList("image/png", "image/jpg", "image/jpeg", "image/gif");
    public static final long MAX_BYTES= 524288;//512 KB de tamaño máximo
    @Override
    public void initialize(ImageValida constraintAnnotation) {
    }
    @Override
    public boolean isValid(MultipartFile multipartFile, ConstraintValidatorContext context) {
        //Por defecto resultado de la comprobación válido hasta que encontremos un error
        boolean result = true;
        //comprobar lista de errores
        ArrayList<String> listaErrores=mensajesErrorImagen(multipartFile);
        //Si hay errores añadirlos al contexto
        if (!listaErrores.isEmpty()){
            context.disableDefaultConstraintViolation();
            //iteramos por la lista de errores para añadirlos al contexto
            for(String textoError:listaErrores) {
                context.buildConstraintViolationWithTemplate(
                    textoError)
                    .addConstraintViolation();
            }
            //Comprobación incorrecta (resultado no valido)
            result = false;
        }
        //Devolvemos resultado de la comprobación
        return result;
    }
}
```



UD 2: Modelo Vista Controlador

10.- Ficheros

Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com



4º Crear un validador personalizado de imágenes de usuarios (anotación)

Prueba a intentar añadir un fichero que no es una imagen, no cumple el tamaño o simplemente a darle actualizar sin añadir un fichero y comprueba si indica los errores.

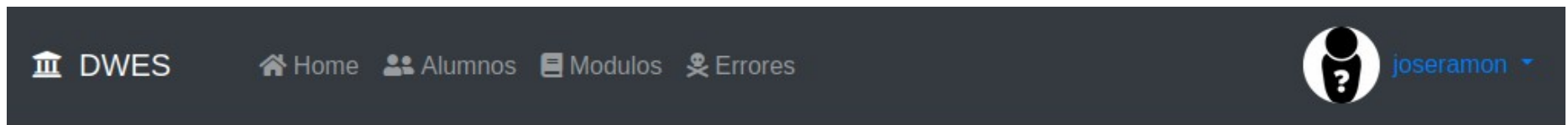


Imagen del usuario:



Puede actualizar la imagen seleccionando una nueva imagen: No se ha seleccionado ningún archivo.

Solo se permiten imagenes PNG, JPG o GIF.

Tamaño máximo de la imagen excedido (524288 bytes)



UD 2: Modelo Vista Controlador

10.- Ficheros

Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com



4º Crear un validador personalizado de imágenes de usuarios (anotación)

Por último prueba a introducir una imagen válida y comprueba que se actualiza correctamente, se guardar en la carpeta y se muestra en la web en la parte superior derecha:









 DWES  Home  Alumnos  Modulos  Errores  joseramon ▾

Imagen del usuario:



Puede actualizar la imagen seleccionando una nueva imagen: No se ha seleccionado ningún archivo.



Cuando apagas el Tomcat y vuelves a encenderlo se resetean los usuarios, pero las imágenes siguen estando en la carpeta correspondiente. **Igual seria buena idea crear un constructor nuevo para la clase 'Usuario' donde le puedas especificar directamente la imagen también y así poder arrancar con la imagen por defecto...**



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

Ya sabemos como añadir imágenes de manera dinámica para cambiar la foto de perfil del usuario. A continuación veremos como podemos **subir al servidor y descargar del servidor ficheros de distintos tipos** para poder incorporarlos a la documentación del alumno :

 DWES

 Home

 Alumnos

 Modulos

 Errores

 [joseramon](#)

Documentación del alumno:

Dni	Nombre	Ciclo	Curso
11111111A	Jose	DAM	1

Si desea añadir nueva documentación introduzca los datos:

Id:

Comentario:

Tipo:


☐ Certificado

☐ Justificante

☐ Solicitud

 Añadir

Seleccione el archivo: No se ha seleccionado ningún archivo.

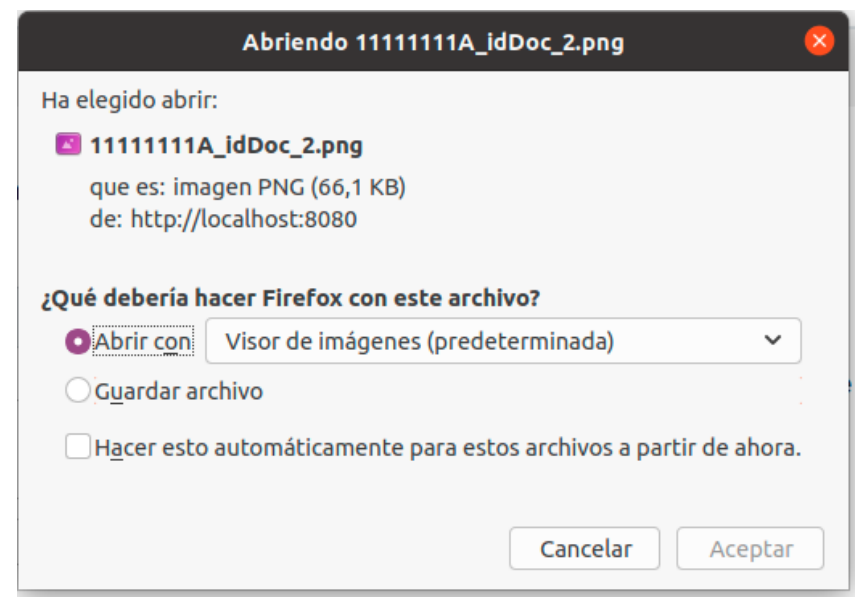
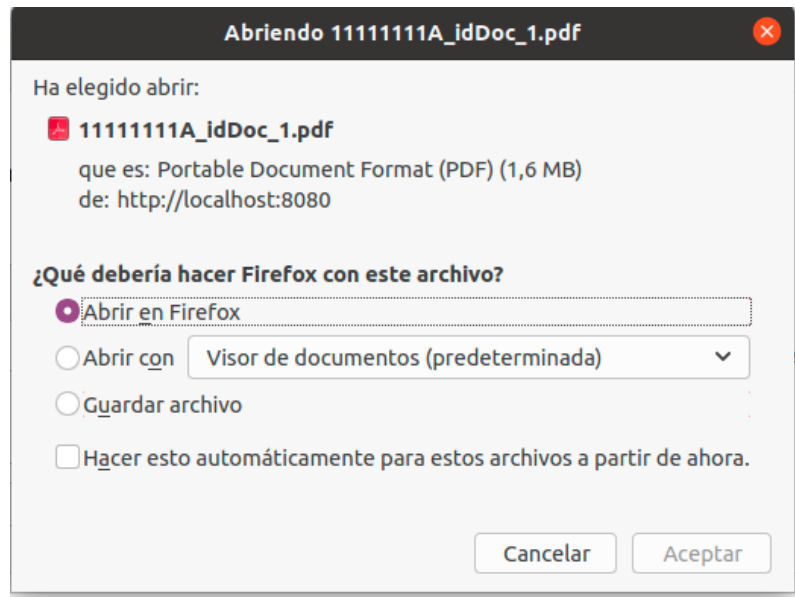
Id	Tipo	Comentario	
1	Certificado (pdf)	Certificado en PDF	 Descargar
2	Justificante (png)	Captura de pantalla del justificante	 Descargar



UD 2: Modelo Vista Controlador

10.- Ficheros

Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com



Piensa que pasos necesitarías hacer para conseguirlo...



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

Consejos:

1º Nueva estructura de DocAlumno para almacenar el fichero:

Deberá de incorporar atributos nuevos: fichero (tipo multipart) , tipoFichero (para poder almacenar la extensión sin tener que extraerla del multipart y poder mostrarla entre paréntesis después del tipo de documento en el listado), contentTypeFichero(para saber el contentType cuando descarguemos el fichero y que el navegador web sea capaz de abrirlo con la aplicación adecuada).

Deberemos de validar solo el “fichero” multipart porque el tipoFichero y contentTypeFichero no lo rellenamos en la vista, sino que lo rellenamos en el controller cuando tenemos el fichero y podemos comprobar esos datos.

```
DocAlumno.java x
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.model;
2
3 import javax.validation.constraints.NotNull;
4
5
6
7
8
9
10 public class DocAlumno implements Comparable<DocAlumno>{
11     @Pattern(regexp = "[0-9]{8}[A-Za-z]{1}", message = "El dni debe tener 8 números y una letra")
12     private String dni;
13     @NotNull(message = "El id no puede estar vacio")
14     private Integer id;
15     @NotNull(message = "El tipo no puede estar vacio")
16     private String tipo;
17     @Size(min = 10, message = "Los comentarios debe tener almenos 10 caracteres")
18     private String comentario;
19
20     @NotNull
21     @DocumentoAlumnoValido
22     private MultipartFile fichero;
23
24     String tipoFichero="";
25     String contentTypeFichero="";
26 }
```



Para realizar la validación crear una clase DocumentoAlumnoValido y otra ValidadorDocumentoAlumno dentro del paquete de validaciones para permitir utilizar la notación @DocumentoAlumnoValido en el bean DocAlumno.

Para añadir un nuevo fichero no valdrá cualquier tipo de documento. Las extensiones del validador son diferentes a las imagenes:

```
ValidadorDocumentoAlumno.java ×  
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.validaciones;  
2  
3 import org.springframework.web.multipart.MultipartFile;  
11  
12 public class ValidadorDocumentoAlumno implements ConstraintValidator<DocumentoAlumnoValido, MultipartFile> {  
13     public static final List<String> tiposDeDocumentos=Arrays.asList("image/png", "image/jpg", "image/jpeg",  
14         "image/gif","text/plain","application/pdf","application/msword","application/vnd.ms-excel",  
15         "application/vnd.openxmlformats-officedocument.wordprocessingml.document","application/excel",  
16         "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",  
17         "application/vnd.oasis.opendocument.text","application/vnd.oasis.opendocument.spreadsheet",  
18         "application/x-rar-compressed","application/zip",  
19         "application/x-zip-compressed","multipart/x-zip");  
20     public static final long MAX_BYTES= 2097152;//2MB de tamaño máximo
```

Adicionalmente el tamaño máximo permitido para los documentos será de 2 Mb.

Piensa como deberias hacer el resto del validador de manera similar al validador de Imagenes. Si necesitas ayuda a continuación tienes el código, pero intenta entenderlo por si sale un validador en el próximo examen ...



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

//Este trozo de código puede copiarse desde el PDF proporcionado al alumno y ocupa esta y la SIGUIENTE DIAPOSITIVA:

```
public class ValidadorDocumentoAlumno implements ConstraintValidator<DocumentoAlumnoValido, MultipartFile> {
    public static final List<String> tiposDeDocumentos=Arrays.asList("image/png", "image/jpg", "image/jpeg",
        "image/gif", "text/plain", "application/pdf", "application/msword", "application/vnd.ms-excel",
        "application/vnd.openxmlformats-officedocument.wordprocessingml.document", "application/excel",
        "application/vnd.openxmlformats-officedocument.spreadsheetml.sheet",
        "application/vnd.oasis.opendocument.text", "application/vnd.oasis.opendocument.spreadsheet",
        "application/x-rar-compressed", "application/zip",
        "application/x-zip-compressed", "multipart/x-zip");

    public static final long MAX_BYTES= 2097152;//2MB de tamaño máximo

    @Override
    public void initialize(DocumentoAlumnoValido constraintAnnotation) {

    }

    @Override
    public boolean isValid(MultipartFile multipartFile, ConstraintValidatorContext context) {
        //Por defecto resultado de la comprobación válido hasta que encontremos un error
        boolean result = true;

        //comprobar lista de errores
        ArrayList<String> listaErrores=mensajesErrorDocumento(multipartFile);
        //Si hay errores añadirlos al contexto
        if (!listaErrores.isEmpty()){
            context.disableDefaultConstraintViolation();
            //iteramos por la lista de errores para añadirlos al contexto
            for(String textoError:listaErrores) {
                context.buildConstraintViolationWithTemplate(
                    textoError)
                    .addConstraintViolation();
            }
            //Comprobación incorrecta (resultado no valido)
            result = false;
        }
        //Devolvemos resultado de la comprobación
        return result;
    }
}
```



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

```
public static ArrayList<String> mensajesErrorDocumento(MultipartFile fichero) {
    ArrayList<String> errores=new ArrayList<String>();
    //Fichero no vacio
    if (fichero==null || fichero.isEmpty()) {
        errores.add("Proporcione un fichero válido.");
        return errores;
    }
    //Validar tipo de fichero
    String contentType = fichero.getContentType();
    //System.out.println("Tipo de documento de '"+fichero.getName()+"': "+contentType);
    if (!tipoDeDocumentoValido(contentType)) {
        errores.add("Tipo de documento '"+contentType+"' no permitido.");
    }
    //Comprobar tamaño máximo
    if (fichero.getSize()>MAX_BYTES) {
        errores.add("Tamaño máximo del documento excedido (" +MAX_BYTES+" bytes)");
    }
    return errores;
}

//Comprueba si el tipo de imagen es uno de los predefinidos en "tiposDeDocumentos"
private static boolean tipoDeDocumentoValido(String contentType) {
    return tiposDeDocumentos.contains(contentType);
}

/** METODOS PUBLICOS PARA UTILIZAR EN OTRAS CLASES, POR EJEMPLO FileService.java */
public static boolean documentoValido(MultipartFile fichero) {
    ArrayList<String> listaErrores=mensajesErrorDocumento( fichero);
    //Si esta vacia no hay errores
    return listaErrores.isEmpty()?true:false;
}

//Comprueba si el tipo de fichero està en la lista "tiposDeDocumentos" y
//devuelve solo la extensión. Si no està devuelve ""
public static String getExtension(MultipartFile fichero) {
    String result="";
    for(String tipoValido:tiposDeDocumentos) {
        if (fichero.getContentType().equals(tipoValido)) //por ejemplo: si es "image/jpg" devolverá "jpg"
            return tipoValido.split("/")[1];
    }
    return result;
}
}
```



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

2º Subir el fichero con la documentación del alumno al servidor web:

```
@RequestMapping(value = "/add-docAlumno", method = RequestMethod.POST)  
public String addDocAlumno(ModelMap model, @Valid DocAlumno docAlumno, BindingResult validacion) {
```

Aunque a continuación se muestra el código debemos entender lo que vamos a hacer:

- Obtener la extensión del documento adjuntado llamando a la función `getExtensionMultipartFile(multipartfile)` del servicio `fileServicio`. No nos servirá el `getExtension()` del validador porque hay `contentType`s que al final no te dice la extensión. En `getExtensionMultipartFile()` deberemos de utilizar el método `getOriginalFilename` del objeto `multipartfile` para buscar el último punto (".") y devolver lo que hay a la derecha.

- Rellenar el nombre del fichero a guardar con el formato "DNI_idDoc_ID.EXT", donde proporcionaremos en variables DNI, ID y EXT.

- Guardar el documento llamando a la función `guardaDocumentacionAlumno` (con los parámetros `multipartFile` y `nombreFichero`) de `fileServicio`.

- Avisar de los errores al guardar si los hay (y salirse) o modificar en el `docAlumno` el tipo de fichero y el `contentType` del fichero.



2º Subir el fichero con la documentación del alumno al servidor web:

//Este trozo de código puede copiarse desde el PDF proporcionado al alumno y ocupa esta y la SIGUIENTE DIAPOSITIVA:

```
@RequestMapping(value = "/add-docAlumno", method = RequestMethod.POST)
public String addDocAlumno(ModelMap model, @Valid DocAlumno docAlumno, BindingResult validacion) {
    paginaServicio.setPagina(pagina);
    model.addAttribute("pagina", paginaServicio.getPagina());
    if (validacion.hasErrors()) {
        // Hay errores y debemos volver al formulario
        //Si no añadimos alumno, la cabecera de los datos del alumno no se imprimiran
        model.addAttribute("alumno", alumnoService.encontrarAlumnoPorDni(docAlumno.getDni()));
        return "docs-alumno";
    }
    // Si llega aquí no hay errores de validación
    String dni = (String) docAlumno.getDni();
    Alumno alumno= alumnoService.encontrarAlumnoPorDni(dni);
    try {
        if (alumno == null) {
            throw new Exception("Alumno desconocido");
        }
        if (model.getAttribute("usuario") == null) {
            throw new Exception("Para añadir documentación debe estar logeado");
        }
        // Guardar fichero en el Sistema Operativo:
        //1º Componer nombre del fichero
        String extension=fileServicio.getExtensionMultipartfile(docAlumno.getFichero());
        String nombreFicheroAGuardar=String.format("%s_idDoc_%s.%s",
            dni,
            docAlumno.getId(),
            extension);

        //2º Guardarlo
        //Si no se ha podido listaErroresAlGuardar no estará vacío
        ArrayList<String>
        listaErroresAlGuardar=fileServicio.guardaDocumentacionAlumno(docAlumno.getFichero(),nombreFicheroAGuardar);
        if(!listaErroresAlGuardar.isEmpty()) {
            //Rellenar los errores al intentar guardar para pasarselos a la excepcion
            String mensajeCompleto="";
```


UD 2: Modelo Vista Controlador

10.- Ficheros

Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com



```

        for(String mensaje:listaErroresAlGuardar) {
            mensajeCompleto+=mensaje+"<br>";
        }
        //lanzar excepción
        throw new Exception(mensajeCompleto);
    }
    //completar tipo de documento del alumno
    docAlumno.setTipoFichero(extension);
    docAlumno.setContentTypeFichero(docAlumno.getFichero().getContentType());
    //Añadir el nuevo documento al alumno :
    alumnoService.addDocAlumno(alumno,docAlumno);//Debe añadir un doc al ArrayList de ese alumno
    Usuario usuarioActivo = (Usuario) model.getAttribute("usuario");
    //Al modificar el alumno (su lista de docs) debemos actualizar la fecha y usuario
    alumnoService.modificaAlumno(alumno, usuarioActivo.getNickname());
    //Si no añadimos alumno, la cabecera de los datos del alumno no se imprimiran
    model.addAttribute("alumno",alumnoService.encontrarAlumnoPorDni(docAlumno.getDni()));
    //Al igual que en el GET, debemos crear un doc vacío con el siguiente id
    model.addAttribute("docAlumno",new DocAlumno(alumnoService.siguienteDoc(dni)));
    // Para evitar pasar parámetros innecesarios
    return "docs-alumno";
} catch (Exception e) {
    // Le pasamos el alumno actualizado
    model.addAttribute(alumnoService.encontrarAlumnoPorDni(alumno.getDni()));
    // Pasamos los errores
    model.addAttribute("errores", e.getMessage());
    // Hay errores y debemos volver al formulario de modificación
    return "docs-alumno";
}
}
    
```



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseamon.profesor@gmail.com

3º Descargar el fichero con el documento del alumno:

Para descargar el documento no proporcionamos el fichero al navegador web como si fuera una imagen, por lo que se suministra el código entero de dicho método en la página siguiente.

Sin embargo el alumno deberá de crear 2 métodos:

Servicio de los alumnos: Método encontrarDocAlumnoPorDni_IdDoc con los parámetros dni e idDoc. Intentar utilizar streams ...

Servicio de los ficheros: Método getDocumentoAlumno(String fichero). A imagen y semejanza de getImagenUsuario(String fichero) ...



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

Método en AlumnoController.java:

//Este trozo de código puede copiarse desde el PDF proporcionado al alumno!!

```
@RequestMapping(value = "/descargar-docAlumno/{dni}/{idDoc}" , method = RequestMethod.GET)
public @ResponseBody void descargarDocAlumno(HttpServletRequest response,@PathVariable("dni") String dni,
    @PathVariable("idDoc") Integer idDoc) throws IOException{

    try {
        Optional<DocAlumno> docAlumno= alumnoService.encontrarDocAlumnoPorDni_IdDoc(dni, idDoc);
        if (docAlumno.isPresent()) {//existe el documento de ese alumno con ese id
            //Obtener fichero
            String nombreFichero=dni+"_idDoc_"+idDoc+"."+docAlumno.get().getTipoFichero();
            FileSystemResource resource = fileServicio.getDocumentoAlumno(nombreFichero);
            if (!resource.exists()) {
                throw new IOException("El documento con el dni '"+dni+"' y el id '"+idDoc+"' no existe;");
            }
            File fichero= resource.getFile();
            //Montar respuesta para el navegador web
            response.setContentType(docAlumno.get().getContentTypeFichero());
            response.setHeader("Content-Disposition", "attachment; filename=" + fichero.getName());
            response.setHeader("Content-Length", String.valueOf(fichero.length()));
            InputStream in = new FileInputStream(fichero);
            FileCopyUtils.copy(in, response.getOutputStream());

        } else {
            throw new IOException ("El documento con el dni '"+dni+"' y el id '"+idDoc+"' no existe;");
        }

    } catch (Exception e) {//Ante cualquier error
        //Devolver error 404-recurso no encontrado
        throw new ResponseStatusException(HttpStatus.NOT_FOUND);
    }

}
```



UD 2: Modelo Vista Controlador

10.- Ficheros



Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

EJERCICIO:

Sube la aplicación final al moodle.

Para ello:

1º Haz un “Run As \Maven Clean” para dejar solo los fichero fuentes y quitar momentaneamente los necesarios para ejecutar la aplicación (dependencias).

2º Comprime la carpeta de tu aplicación y ponle como nombre al fichero comprimido UD2_practica8_nombreAlumno.tar.gz donde nombreAlumno es el nombre del alumno que entrega la práctica.

3º Súbela al moodle.

IMPORTANTE: No comprimir en RAR, porque Ubuntu no lo lee bien y en clase tenemos Ubuntu. Si tuviesemos Windows, podemos comprimir en ZIP.