



UD 2: Modelo Vista Controlador Ayuda 'DocAlumnos' y 'Filtro Avanzado':

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



El alumno debería intentar hacer el ejercicio de consolidación de Aules relativo a “DocAlumnos” y el “Filtro Avanzado” sin ayuda en 3 horas.

De hecho , este ejercicio de consolidación es un antiguo examen de la UD2 hasta lo visto en Streams.

Si por el contrario, todavía no tienes claro como proceder, a continuación se explica los pasos a realizar con cierto nivel de detalle.



Partiendo de la última práctica entregada Streams vamos a realizar los siguientes cambios a la aplicación:

Deberemos añadir al listado de alumnos un nuevo botón para **VISUALIZAR LA DOCUMENTACIÓN DE CADA UNO DE LOS ALUMNOS:**

Simarro Home **Alumnos** Modulos Errores DWES Logout

Listado de alumnos:

Opciones: Dni Como: Filtrar

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

[Añadir alumno](#)

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

Simarro Home **Alumnos** Modulos Errores DWES Logout

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](#)

Id	Tipo	Comentario
1	Justificante	Justificante alumno 11111111A
2	Certificado	Certificado alumno 11111111A



¿Por donde empezamos?



¿Como llegamos a la página de documentación?

El botón de “Documentación” de un alumno funcionará de manera similar a como funciona el botón de “Modificar”, esto significa que llamará a una url atendida por el controller con un parámetro “dni” para saber de que alumno estamos hablando:

```
<a class="btn btn-success" href="docs-alumno?dni=${alumno.getDni()}">Documentación</a>
```

¿Como guardaremos la información?

Hay que tener claro que cada alumno tiene su ArrayList de sus documentaciones :

```
DocAlumno.java
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.model;
2
3 import javax.validation.constraints.NotNull;
4
5
6
7 public class DocAlumno implements Comparable<DocAlumno>{
8     @Pattern(regexp = "[0-9]{8}[A-Za-z]{1}", message = "El dni debe te
9     private String dni;
10    @NotNull(message = "El id no puede estar vacio")
11    private Integer id;
12    @NotNull(message = "El tipo no puede estar vacio")
13    private String tipo;
14    @Size(min = 10, message = "Los comentarios debe tener almenos 10 c
15    private String comentario;
16
17
18    public String getDni() {
19        return dni;
20    }
21    public void setDni(String dni) {
22        this.dni = dni;
23    }
24    public Integer getId() {
25        return id;
26    }
27    public void setId(Integer id) {
28        this.id = id;
29    }
30 }
```

```
Alumno.java
16
17 public class Alumno implements Modificable<Alumno>,Serializable, Comparable<Alumno>{
18     private static final long serialVersionUID = 1L;
19     @Pattern(regexp = "[0-9]{8}[A-Za-z]{1}", message = "El dni debe tener 8 números y una letra")
20     private String dni;
21     @Size(min=5,message="El nombre debe de tener un tamaño mínimo de 5 caracteres")
22     private String nombre;
23     @NotNull(message = "La edad no puede estar vacia")
24     @Range(min = 18, max = 99, message = "La edad debe ser igual o mayor a 18 y menor o igual a 99")
25     @Digits(integer = 2,fraction = 0, message = "La edad no puede tener decimales ni más de 2 dígitos")
26     private Integer edad;
27     @Size(min = 3, message = "El ciclo debe tener almenos 3 caracteres")
28     private String ciclo;
29     @NotNull(message = "El curso no puede estar vacio")
30     @Digits(fraction = 0, integer = 1,message = "El curso tiene un formato incorrecto")
31     @Range(min = 1, max = 2, message = "El curso solo admite los valores 1 o 2")
32     private Integer curso;
33
34     private boolean erasmus=false;
35     private String[] interesadoEn;
36     private String lenguajeFavorito="";
37     private String genero;
38     private String horario;
39     private String pais;
40     private ArrayList<Integer> matriculadoEn;
41     private String hobbies;
42     private ArrayList<DocAlumno> docsAlumno;
43     private Date ts;
44     private String user;
```



UD 2: Modelo Vista Controlador Ayuda 'DocAlumnos' y 'Filtro Avanzado':



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

¿Como montamos la nueva página?

Teniendo claro como se guarda la información, el nuevo JSP tiene 3 partes claramente diferenciadas:

1º Información del alumno

2º Formulario de alta de documentación

3º Listado de documentación de un alumno en concreto



... continuación ¿Como montamos la nueva página?

En base a esta información, **docs-alumno.jsp** podría tener el siguiente aspecto:

Información del alumno: **Formulario de alta de la documentación:**

docs-alumno.jsp

```
1 <%@ include file="../../jspf/header.jspf"%>
2 <%@ include file="../../jspf/menuSuperior.jspf"%>
3
4 <div class="container">
5     <h1> Documentación del alumno:</h1>
6     <table class="table table-striped">
7         <thead>
8             <th> Dni</th>
9             <th> Nombre</th>
10            <th> Ciclo</th>
11            <th> Curso</th>
12        </thead>
13        <tbody>
14            <tr>
15                <td>&nbsp;${alumno.getDni()}&nbsp;</td>
16                <td>&nbsp;${alumno.getNombre()}&nbsp;</td>
17                <td>&nbsp;${alumno.getCiclo()}&nbsp;</td>
18                <td>&nbsp;${alumno.getCurso()}&nbsp;</td>
19            </tr>
20        </tbody>
21    </table>
22    <br>
```

```
23 <p> Si desea añadir nueva documentación introduzca los datos:</p>
24 <p>
25     <font color="red">${errores}</font>
26 </p>
27
28 <Mvc:form method="post" action="add-docAlumno" modelAttribute="docAlumno">
29
30     <Mvc:errors path="*" cssClass="text-warning" />
31     <input name="dni" type="hidden" value="${alumno.getDni()}">
32     <div class="form-row">
33         <div class="col">
34             <Mvc:label path="Id">Id:</Mvc:label>
35             <Mvc:input path="id" type="text" readonly="true" class="form-control"/>
36             <Mvc:label path="comentario">Comentario:</Mvc:label><br>
37             <Mvc:textarea path="comentario" rows="2" cols="70" />
38         </div>
39         <div class="col">
40             <Mvc:label path="tipo">Tipo:</Mvc:label>
41             <ul>
42                 <Mvc:radiobuttons path="tipo" items="${opcionesTipoDoc}" element="p" />
43             </ul>
44         </div>
45         <div class="col">
46             <br>
47             <br>
48             <input type="submit" value="Añadir" class="btn btn-success">
49         </div>
50     </div>
51
52 </Mvc:form>
```

En la sección "Información del alumno" por falta de tiempo se aceptaría copy&paste del código del listado de alumnos, siempre y cuando hubieramos creado una lista solo con ese único alumno, pero realmente hubiera sido una chapucilla.

En **cualquier formulario** hay que tener en cuenta que SIEMPRE se debe pasar el **modelAttribute** y debe ser un bean con el mismo nombre que la clase.



UD 2: Modelo Vista Controlador Ayuda 'DocAlumnos' y 'Filtro Avanzado':



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

... continuación **¿Como montamos la nueva página?**

Y la parte final del JSP que muestra el listado de documentación del alumno:

Listado de documentación de un alumno:

```
53
54     <br>
55     <table class="table table-striped">
56         <thead>
57             <th> Id</th>
58             <th> Tipo</th>
59             <th> Comentario</th>
60         </thead>
61         <tbody>
62             <c:forEach items="${alumno.getDocsAlumno()}" var="docAlumno">
63                 <tr>
64                     <td>&nbsp;${docAlumno.getId()}&nbsp;</td>
65                     <td>&nbsp;${docAlumno.getTipo()}&nbsp;</td>
66                     <td>&nbsp;${docAlumno.getComentario()}&nbsp;</td>
67                 </tr>
68             </c:forEach>
69         </tbody>
70     </table>
71 </div>
72
73 <%@ include file="../jspf/footer.jspf"%>
74
```



UD 2: Modelo Vista Controlador

Ayuda 'DocAlumnos' y 'Filtro Avanzado':



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

... continuación **¿Como montamos la nueva página?**

Para rellenar el código del **controller** debemos tener en cuenta la información que necesita la vista:

1º Información del alumno:

Debemos pasar a través del modelo el objeto “alumno” correspondiente a ese dni utilizando el método de alumnoService que nos devuelva un alumno en base a su dni.

2º Formulario de alta de documentación:

Debemos de pasar al modelo un objeto de tipo “documentacionAlumno” cuyo id este precalculado en base a un método que nos diga el siguiente id de documentación de ese alumno (si no existe devolverá 1).

3º Listado de documentación de un alumno en concreto:

Ya forma parte del alumno, por lo que no hace falta pasar nada.

```
@RequestMapping(value = "/docs-alumno", method = RequestMethod.GET)
public String getDocsAlumnos(ModelMap model, @RequestParam String dni) {

    paginaServicio.setPagina(pagina);
    model.addAttribute("alumno", alumnoService.encontrarAlumnoPorDni(dni));
    model.addAttribute("docAlumno", new DocAlumno(alumnoService.siguienteDoc(dni)));
    model.addAttribute("pagina", paginaServicio.getPagina());
    return "docs-alumno";
}
```

Debes implementar el método “siguienteDoc”



UD 2: Modelo Vista Controlador Ayuda 'DocAlumnos' y 'Filtro Avanzado':



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

¿Como añadimos una documentacion nueva?

Desde el formulario se puede llamar a “add-doc-alumno” o incluso a “docs-alumnos” en modo POST para añadir datos. Debemos tener en cuenta que el usuario esté logeado porque deberemos actualizar la fecha modificación del alumno.

Una forma más limpia seria controlar la excepción de “Alumno desconocido” y “Usuario logeado” dentro del “addDocAlumno” del service, en vez de en el controller.

Se ha hecho en el controller para que el alumno vea las condiciones a controlar.

```
AlumnoController.java
140 @RequestMapping(value = "/add-docAlumno", method = RequestMethod.POST)
141 public String addDocAlumno(ModelMap model, @Valid DocAlumno docAlumno, BindingResult validacion) {
142     paginaServicio.setPagina(pagina);
143     model.addAttribute("pagina", paginaServicio.getPagina());
144     if (validacion.hasErrors()) {
145         // Hay errores y debemos volver al formulario
146         // Si no añadimos alumno, la cabecera de los datos del alumno no se imprimiran
147         model.addAttribute("alumno", alumnoService.encontrarAlumnoPorDni(docAlumno.getDni()));
148         return "docs-alumno";
149     }
150     // Si llega aquí no hay errores de validación
151     String dni = (String) docAlumno.getDni();
152     Alumno alumno = alumnoService.encontrarAlumnoPorDni(dni);
153     try {
154
155         if (alumno == null) {
156             throw new Exception("Alumno desconocido");
157         }
158         if (model.getAttribute("usuario") == null) {
159             throw new Exception("Para añadir documentación debe estar logeado");
160         }
161         alumnoService.addDocAlumno(alumno, docAlumno); // Debe añadir un doc al ArrayList de ese alumno
162         Usuario usuarioActivo = (Usuario) model.getAttribute("usuario");
163         // Al modificar el alumno (su lista de docs) debemos actualizar la fecha y usuario
164         alumnoService.modificaAlumno(alumno, usuarioActivo.getNickname());
165         // Si no añadimos alumno, la cabecera de los datos del alumno no se imprimiran
166         model.addAttribute("alumno", alumnoService.encontrarAlumnoPorDni(docAlumno.getDni()));
167         // Al igual que en el GET, debemos crear un doc vacío con el siguiente id
168         model.addAttribute("docAlumno", new DocAlumno(alumnoService.siguienteDoc(dni)));
169         // Para evitar pasar parámetros innecesarios
170         return "docs-alumno";
171     } catch (Exception e) {
172         // Le pasamos el alumno actualizado
173         model.addAttribute(alumnoService.encontrarAlumnoPorDni(alumno.getDni()));
174         // Pasamos los errores
175         model.addAttribute("errores", e.getMessage());
176         // Hay errores y debemos volver al formulario de modificación
177         return "docs-alumno";
178     }
179 }
```




¿Que efecto secundario podemos tener?

Como se indica en el enunciado, si después de añadir una documentación vamos a modificar un alumno, como en el formulario de modificación “no tenemos ni debemos mostrar la documentación” sin darnos cuenta machacaremos un alumno con una lista de documentaciones por el nuevo alumno modificado sin lista de documentaciones (al no estar en el formulario de modificación).

Hasta que veamos DTOs, la solución pasa por actualizar el usuario modificado con la lista de documentaciones antes de machacarlo:

AlumnoService.java

```
104 public void modificaAlumno(Alumno alumnoModificado,String usuarioModificacion) throws Exception {
105     String errores="";
106     if (alumnoModificado==null) {
107         errores="No se ha podido actualizar el alumno porque no han llegado los datos modificados";
108     } else {
109         Alumno alumnoActual=encontrarAlumnoPorDni(alumnoModificado.getDni());
110         if (alumnoActual.sePuedeModificarUtilizando(alumnoModificado)) {
111             alumnos.remove(alumnoActual);
112             //Debemos de asegurarnos que no se borra la documentación
113             alumnoModificado.setDocsAlumno(alumnoActual.getDocsAlumno());
114             //actualizamos usuario y fecha modificación
115             alumnoModificado.setUser(usuarioModificacion);
116             alumnoModificado.setTs(new Date());
117             alumnos.add(alumnoModificado);
118         } else {
119             errores=alumnoActual.mensajeNoSePuedeModificar();
120         }
121     }
122     if (errores.length()>0) {
123         throw new Exception(errores);
124     }
125 }
```



UD 2: Modelo Vista Controlador

Ayuda 'DocAlumnos' y 'Filtro Avanzado':

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



¿Como implementamos el nuevo filtro?

Primero es quitar el filtro anterior e introducir el nuevo: (Habrá que crear la clase FiltroAvanzadoAlumno.java)

```
list-alumno.jsp
5      <h1> Listado de alumnos:</h1>
6      <mvc:form method="post" action="filtro-avanzado-alumnos" modelAttribute="filtroAvanzadoAlumno">
7      <div class="form-row">
8          <div class="col">
9              <mvc:label path="dni">Dni:</mvc:label>
10             </div>
11             <div class="col">
12                 <mvc:label path="ciclo">Ciclo:</mvc:label>
13             </div>
14             <div class="col">
15                 <mvc:label path="horario">Horario:</mvc:label>
16             </div>
17             <div class="col">
18                 &nbsp;
19             </div>
20         </div>
21         <div class="form-row">
22
23             <div class="col">
24                 <mvc:select path="dni">
25                     <mvc:option value="-" label="Ninguno" />
26                     <mvc:options items="${dniListaAlumnos}" />
27                 </mvc:select>
28             </div>
29             <div class="col">
30                 <mvc:select path="ciclo">
31                     <mvc:option value="-" label="Ninguno" />
32                     <mvc:options items="${cicloListaAlumnos}" />
33                 </mvc:select>
34             </div>
35             <div class="col">
36                 <mvc:select path="horario">
37                     <mvc:option value="-" label="Ninguno" />
38                     <mvc:options items="${horarioListaAlumnos}" />
39                 </mvc:select>
40             </div>
41             <div class="col">
42                 <input type="submit" class="btn btn-success" value="Filtrar">
43             </div>
44         </div>
45     </mvc:form>
```



... continuación **¿Como implementamos el nuevo filtro?**

Se muestra solo el código correspondiente a uno de los listados y se deja a los alumnos implementar los demás:

```
AlumnoController.java ✖  
270  
271 @ModelAttribute("cicloListaAlumnos")  
272 public List<String> getCicloListaAlumnos() {  
273     return alumnoService.cicloListaAlumnos();  
274 }
```

```
AlumnoService.java ✖  
270  
277 public List<String> cicloListaAlumnos() {  
278     Set<String> listaSinRepeticiones=alumnos.stream().filter(a->a.getCiclo()!=null).map(a->a.getCiclo()).collect(Collectors.toSet());  
279     return listaSinRepeticiones.stream().sorted().collect(Collectors.toList());  
280 }
```



... continuación ¿Como implementamos el nuevo filtro?

Y cuando pulsamos en filtrar el Controller deberá atender la petición:

AlumnoController.java

```

240
247 @RequestMapping(value = "/filtro-avanzado-alumnos", method = RequestMethod.POST)
248 public String filtroAvanzadoAlumno(@Valid FiltroAvanzadoAlumno filtroAvanzadoAlumnos, BindingResult validacion, ModelMap model) {
249     paginaServicio.setPagina(pagina);
250     model.addAttribute("pagina", paginaServicio.getPagina());
251     model.put("filtroAvanzadoAlumno", filtroAvanzadoAlumnos);
252     if (validacion.hasErrors()) {
253         // Hay errores y debemos volver al formulario con la lista de alumnos completa
254         model.put("alumnos", alumnoService.listaAlumnos());
255     } else {
256         model.put("alumnos", alumnoService.filtroAvanzadoAlumnos(filtroAvanzadoAlumnos));
257     }
258     return "list-alumno";
259 }

```

Y realizar el filtrado: (Se deja a los alumnos filtrar por dni y horario)

AlumnoService.java

```

190
197 public List<Alumno> filtroAvanzadoAlumnos(FiltroAvanzadoAlumno filtroAvanzadoAlumno) {
198     List<Alumno> lista= alumnos; // lista inicialmente contiene todos los alumnos
199     try { // Ahora si falta vamos filtrando y quedandonos con los alumnos resultante del filtrado
200         // Desconocido si hacemos una depuración se almacena como "-"
201         if (!"-".equals(filtroAvanzadoAlumno.getDni())) { // Si que debemos filtrar por dni
202             lista=
203         }
204         if (!"-".equals(filtroAvanzadoAlumno.getCiclo())) { // Si que debemos filtrar por ciclo
205             lista=lista.stream().filter(a-> a.getCiclo()!=null && a.getCiclo().equals(filtroAvanzadoAlumno.getCiclo())).collect(Collectors.toList());
206         }
207         if (!"-".equals(filtroAvanzadoAlumno.getHorario())) { // Si que debemos filtrar por horario
208             lista=
209         }
210     }
211     catch (Exception e) { // Si hay error la lista debe estar vacia
212         lista= new ArrayList<Alumno>();
213     }
214
215     if (lista==null) // Si hay cualquier problema y lista==null devolvemos lista vacia
216         return new ArrayList<Alumno>();
217     return lista; // Devolvemos la lista después de realizar los filtros
218 }

```

```

<mvc:option value="-" label="Ninguno" />
<mvc:options items="${dniListaAlumnos}" />

```