



UD 2: Modelo Vista Controlador

1.- Introducción



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

Objetivos de la sesión:

- Entender la arquitectura web basada en el patrón de diseño **Modelo Vista Controlador (MVC)**.
- Entender el **patrón de diseño modificado “Front Controller” de Spring MVC**.
- Crear nuestra **primera aplicación Spring MVC**:
 - Crear un nuevo proyecto con las dependencias y configuración de Spring MVC
 - Configurar el `DispatcherServlet` que actúa de “Front Controller”
 - Configurar el controlador de login para atender la petición del navegador a la url de login.
 - Configurar la vista (“View Resolver”) para poder mostrar una página JSP con los datos del modelo (`ModelMap`).
 - Inyectar como dependencias los servicios que utilizará el controlador.



Arquitectura web: Modelo 1:

· Hace 15-20 años ya existían diversas tecnologías para desarrollar páginas web (jsp (hechas en Java), php , asp (hechas en Visual Basic Script) , pero en todas esas tecnologías ***las páginas (jsp,php,asp,...) se encargaban de consultar los datos, implementar la lógica de negocio y presentar los datos al usuario:***



¡La gran desventaja de esta arquitectura era la gran dificultad de mantenimiento !



UD 2: Modelo Vista Controlador

1.- Introducción



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

Arquitectura web: Modelo 2: Patrón de diseño MVC

• El patrón de diseño Modelo Vista Controlador (Model View Controller) nos permite ***dividir una aplicación en unidades o capas independientes con una funcionalidad concreta***, lo cual nos permite cumplir el ***principio de separación de responsabilidades***:

Modelo:

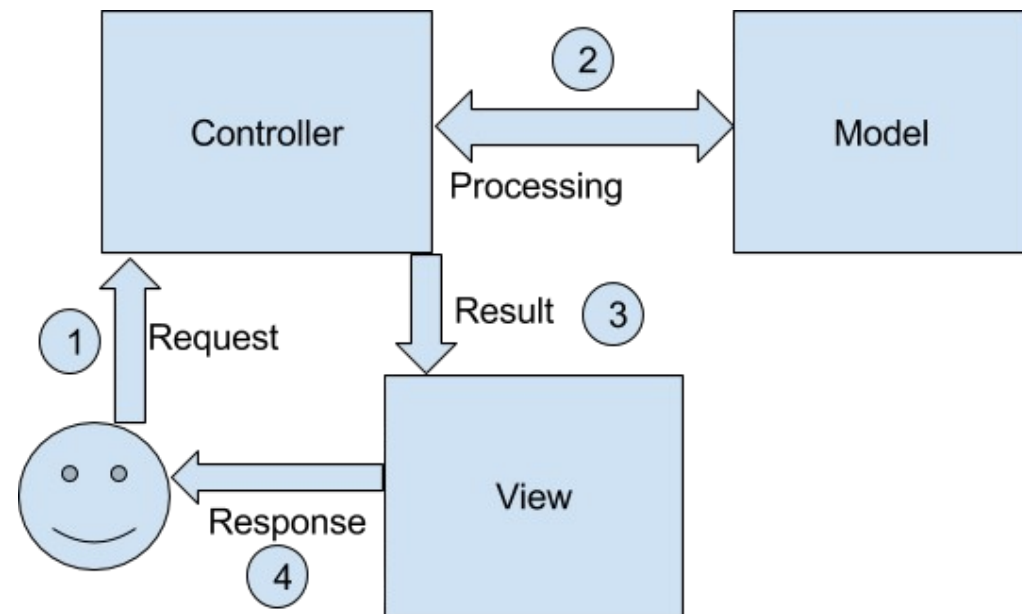
Alumno.java, Modulo.java, LogError.java

Vista:

login.jsp, list-alumno.jsp, add-alumno.jsp

Controlador:

LoginServlet.java, AddAlumnoServlet.java



- El **modelo** es transferido de una capa a otra
- La **vista** es responsable de mostrar los datos de la aplicación al usuario
- El **controlador** es responsable de aceptar una petición del usuario, consultar o modificar el modelo que contiene los datos (utilizando servicios como AlumnoService.java) y reenviar el modelo a la vista.



UD 2: Modelo Vista Controlador

1.- Introducción

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



... continuación **Arquitectura web: Modelo 2: Patrón de diseño MVC**

Al utilizar el patrón de diseño MVC obtenemos varias **ventajas**:

- ***Desarrollo de aplicaciones más rápidas:***

Si definimos la capa del modelo por adelantado y pasamos esta información al equipo de desarrollo de la interfaz gráfica (UI), pueden empezar haciendo las vistas mientras que el equipo de Backend desarrollan los controladores y los servicios.

- ***Mayor flexibilidad ante cambios:***

Si cambiamos el mecanismo utilizado por la vista, para el controlador este cambio es transparente. El controlador solo conoce el nombre de una vista lógica. Por lo tanto, es muy fácil cambiar a una vista basada en velocity o freemaker sin afectar al controlador.

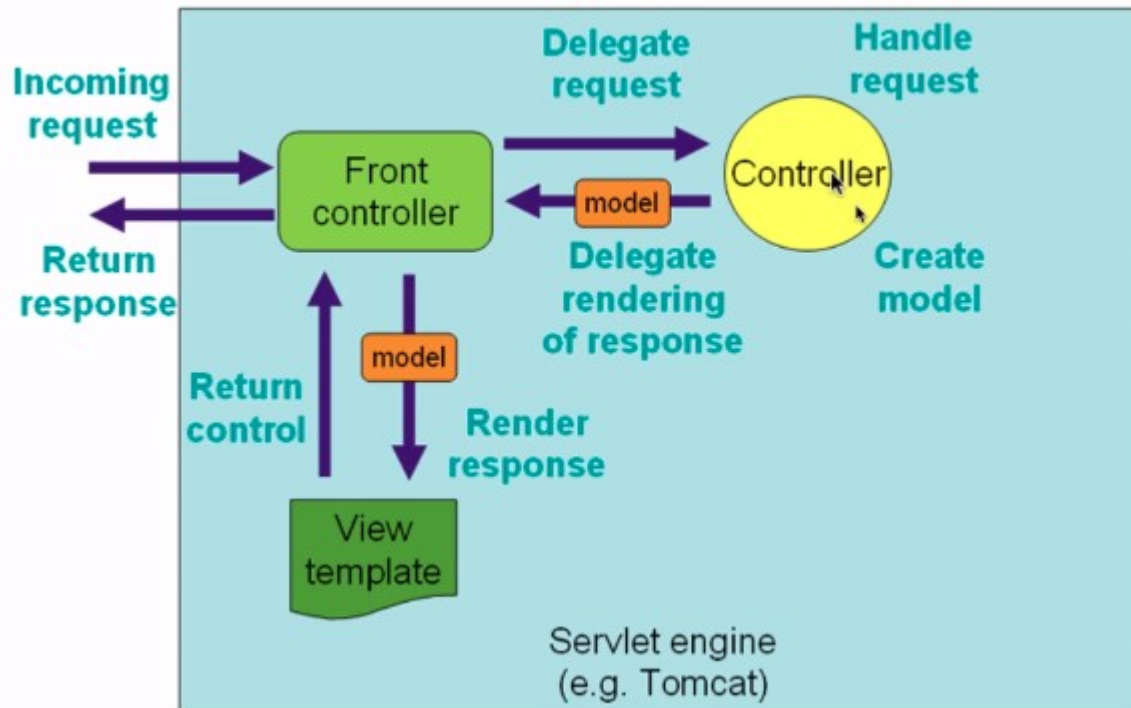
- ***Reducción de tareas de mantenimiento por el bajo acoplamiento:***

La capa de vista es independiente de la capa de servicio. Aunque la capa del controlador depende de la capa de servicio, si se utilizan interfaces tampoco depende de la implementación concreta. Este hecho implica que no hace falta realizar tantos cambios cuando modificamos algo. En el modelo 1 los cambios implicaría realizar tareas de mantenimiento en más de un lugar.



Arquitectura web: Modelo 2 modificado : Patrón de diseño Front Controller

Spring MVC tiene una arquitectura basada en el patrón de diseño MVC , pero algo modificado:



En vez de recibir la petición el controlador correspondiente, **todas las peticiones son preprocesadas por un Servlet especial (el dispatcher Servlet)** que centraliza las peticiones, analiza a quien va dirigida la petición y reenvía dicha petición al controlador correspondiente.



UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador



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

Mi primera aplicación con Spring MVC:

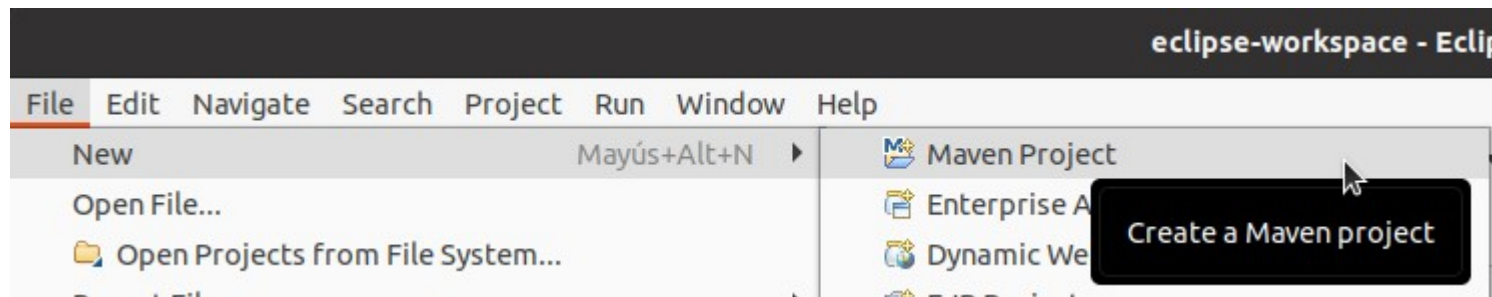
Vamos a crear una nueva aplicación para utilizar el patrón de diseño MVC con los siguientes **10 pasos**:

- 1º Creamos un nuevo proyecto maven sin arquetipo
- 2º Rellenar los datos del nuevo artefacto
- 3º Actualizar el fichero /pom.xml
- 4º Forzamos la descarga de las nuevas dependencias (incluida spring mvc)
- 5º Configurar el dispatcher servlet en src\main\webapp\WEB-INF\web.xml
- 6º Crear el fichero de configuración del contexto de Spring:
src\main\webapp\WEB-INF\alumno-servlet.xml
- 7º Crear las clases Java y los ficheros JSP para hacer el login inicial
- 8º Comprobar que el login funciona
- 9º Crear nuestro primer controlador Spring MVC : LoginController.java
- 10º Comprobar que la web nos da la bienvenida

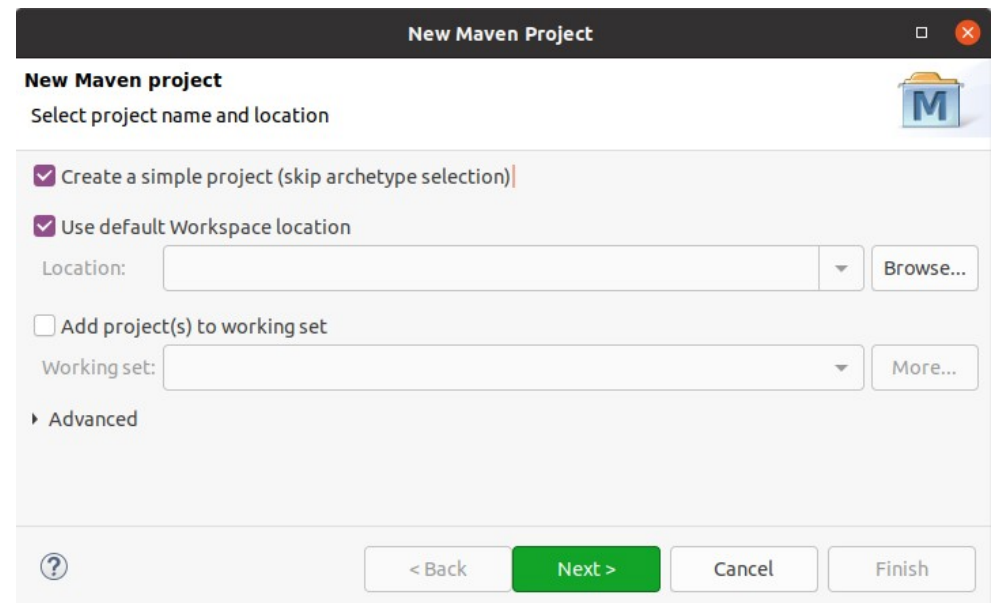


... continuación **Mi primera aplicación con Spring MVC:**

1º Creamos un nuevo proyecto maven: **File \ New \ Maven Project**



Le indicamos que queremos un proyecto simple (**Create a simple project**) que no tenga un arquetipo concreto.





UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador

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



... continuación **Mi primera aplicación con Spring MVC:**

2º Rellenamos los datos del nuevo artefacto:

Es importante **rellenar exactamente con los siguientes datos** para que el profesor pueda cargar correctamente el proyecto de todos los alumnos para su revisión:

Group Id:
org.alumno.nomAlumno
(en minúsculas todo)

Artifact Id:
nomAlumno_primer_app_spring_mvc

Packing:
war

New Maven Project

New Maven project
Configure project

Artifact

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

Parent Project

Group Id:

Artifact Id:

Version:

Browse... Clear

Advanced

? < Back Next > Cancel Finish



UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador



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

... continuación **Mi primera aplicación con Spring MVC:**

3º Actualizar el fichero /pom.xml:

Vamos a **modificar el fichero pom.xml** para dejarlo como en nuestro proyecto de la UD1 copiando todo el contenido por debajo de packaging (dependencias y plugins) : Es normal que “packaging” esté en rojo si no tenemos WEB-INF\web.xml

```
joseramon_primer_app_spring_mvc/pom.xml 25
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2   <modelVersion>4.0.0</modelVersion>
3   <groupId>org.profesor.joseramon</groupId>
4   <artifactId>joseramon_primer_app_spring_mvc</artifactId>
5   <version>0.0.1-SNAPSHOT</version>
6   <packaging>war</packaging>
7   <dependencies>
8     <dependency>
9       <groupId>javax</groupId>
10      <artifactId>javaee-web-api</artifactId>
11      <version>8.0</version>
12      <scope>provided</scope>
13    </dependency>
14    <dependency>
15      <groupId>jstl</groupId>
16      <artifactId>jstl</artifactId>
17      <version>1.2</version>
18    </dependency>
19    <dependency>
20      <groupId>org.webjars</groupId>
21      <artifactId>bootstrap</artifactId>
22      <version>4.5.2</version>
23    </dependency>
24  </dependencies>
25  <build>
26    <pluginManagement>
27      <plugins>
28        <plugin>
29          <groupId>org.apache.maven.plugins</groupId>
30          <artifactId>maven-compiler-plugin</artifactId>
31          <version>3.8.1</version>
32          <configuration>
33            <verbose>true</verbose>
34            <source>1.8</source>
35            <target>1.8</target>
36            <showWarnings>true</showWarnings>
37          </configuration>
38        </plugin>
39        <plugin>
40          <groupId>org.apache.tomcat.maven</groupId>
41          <artifactId>tomcat7-maven-plugin</artifactId>
42          <version>2.2</version>
43          <configuration>
44            <path>/</path>
45            <contextReloadable>true</contextReloadable>
46          </configuration>
47        </plugin>
48      </plugins>
49    </pluginManagement>
50  </build>
51 </project>
```



UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador

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



... continuación **Mi primera aplicación con Spring MVC:**

... continuación 3º Actualizar el fichero /pom.xml:

Ahora vamos a **añadir la dependencia de “spring MVC”** al fichero pom.xml:
Para ello podemos buscar cual es la última dependencia de spring MVC de Maven en Google:

Google search results for "spring mvc maven dependency".

Approximadamente 1.100.000 resultados (0,58 segundos)

mvnrepository.com > artifact > spri... Traducir esta página

org.springframework » spring-webmvc - Maven Repository

... PDF Libraries · Top Categories · Home » org.springframework » spring-webmvc. Spring Web MVC. Spring Web MVC ... Version, Repository, Usages, Date ...

Tags: springmvcwebframework Used By: 3,988 artifacts

Search for groups, artifacts, categories

Spring Web MVC

License: Apache 2.0

Categories: Web Frameworks

Tags: spring, mvc, web, framework

Used By: 4,009 artifacts

Version	Repository	Usages	Date
5.2.9.RELEASE	Central	437	Sep, 2020
5.2.8.RELEASE	Central	447	Jul, 2020
5.2.7.RELEASE	Central	428	Jun, 2020
5.2.6.RELEASE	Central	431	Apr, 2020
5.2.5.RELEASE	Central	446	Mar, 2020
5.2.4.RELEASE	Central	415	Feb, 2020
5.2.3.RELEASE	Central	79	Jan, 2020

```
joseramon_primer_app_spring_mvc/pom.xml

<dependencies>
  <dependency>
    <groupId>javax</groupId>
    <artifactId>javaee-web-api</artifactId>
    <version>8.0</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
  </dependency>
  <dependency>
    <groupId>org.webjars</groupId>
    <artifactId>bootstrap</artifactId>
    <version>4.5.2</version>
  </dependency>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.2.9.RELEASE</version>
  </dependency>
</dependencies>
```



UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador



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

... continuación **Mi primera aplicación con Spring MVC:**

4º Forzamos la descarga de las nuevas dependencias:

Para ello ejecutamos **Run As\Maven clean** y comprobamos en “Library\ Maven Dependencies” si nos hemos descargado las librerías de spring :

The screenshot shows the Eclipse IDE interface. On the left, the 'Project Explorer' shows the 'Java Resources' and 'Libraries' sections. Under 'Libraries', 'Maven Dependencies' is expanded, showing a list of downloaded JAR files including 'javaee-web-api-8.0.jar', 'jstl-1.2.jar', 'bootstrap-4.5.2.jar', 'jquery-3.5.1.jar', 'popper.js-1.16.0.jar', 'spring-webmvc-5.2.9.RELEASE.jar', 'spring-aop-5.2.9.RELEASE.jar', 'spring-beans-5.2.9.RELEASE.jar', 'spring-context-5.2.9.RELEASE.jar', 'spring-core-5.2.9.RELEASE.jar', 'spring-jcl-5.2.9.RELEASE.jar', and 'spring-expression-5.2.9.RELEASE.jar'.

On the right, the 'pom.xml' file is open, showing the following XML content:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.profesor.joseramon</groupId>
    <artifactId>joseramon_primer_app_spring_mvc</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.2.9.RELEASE</version>
        </dependency>
    </dependencies>
</project>
```

Below the XML editor, the 'Console' view shows the output of the 'maven-clean-plugin' goal, indicating that the project was successfully cleaned and built.

```
<terminated> /usr/lib/jvm/java-13-openjdk-amd64/bin/java (25 oct. 2020 19:22:09)
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.SimpleLoggerFactory]
[INFO] Scanning for projects...
[INFO]
[INFO] -----< org.profesor.joseramon:joseramon_primer_app_spring_mvc >-----
[INFO] Building joseramon_primer_app_spring_mvc 0.0.1-SNAPSHOT
[INFO] -----[ war ]-----
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ joseramon_primer_app_s
[INFO] Deleting /home/joseramon/eclipse-workspace/joseramon_primer_app_spring_mv
[INFO]
[INFO] BUILD SUCCESS
[INFO]
```




UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador



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

... continuación **Mi primera aplicación con Spring MVC:**

5º Crear y modificar src/main/webapp/WEB-INF/web.xml:

Si **copiamos el fichero web.xml del proyecto de la UD1** obtenemos el siguiente fichero:

The screenshot shows an IDE with two panels. The left panel, 'Project Explorer', displays the project structure for 'joseramon_primer_app_spring_mvc'. The right panel shows the content of 'web.xml'.

Project Explorer Structure:

- joseramon_primer_app_spring_mvc
 - Deployment Descriptor: joseramon_primer_app_spring_mvc
 - JAX-WS Web Services
 - Java Resources
 - src/main/java
 - src/main/resources
 - src/test/java
 - src/test/resources
 - Libraries
 - Deployed Resources
 - src
 - main
 - java
 - resources
 - webapp
 - WEB-INF
 - alumno-servlet.xml
 - web.xml

web.xml Content:

```
1 <!-- webapp/WEB-INF/web.xml -->
2 <web-app xmlns="http://java.sun.com/xml/ns/javae"
3 <display-name>Lista de alumnos</display-name>
4 <welcome-file-list>
5 <welcome-file>login.do</welcome-file>
6 </welcome-file-list>
7 </web-app>
```




UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador

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



... continuación **Mi primera aplicación con Spring MVC:**

... continuación 5º Crear y modificar src\main\webapp\WEB-INF\web.xml:

Ahora ya podemos **añadir el Dispatcher Servlet** añadiendo el siguiente código a web.xml antes de cerrar el tag <web-app> :

```
<servlet>
  <servlet-name>dispatcher</servlet-name>
  <servlet-class>
    org.springframework.web.servlet.DispatcherServlet
  </servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/alumno-servlet.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>dispatcher</servlet-name>
  <url-pattern>/spring-mvc/*</url-pattern>
</servlet-mapping>
```

UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador



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

... continuación **Mi primera aplicación con Spring MVC:**

... continuación 5º Crear y modificar src\main\webapp\WEB-INF\web.xml:

El fichero web.xml debería quedar de la siguiente manera:

```

1 <!-- webapp/WEB-INF/web.xml -->
2 <web-app xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http:
3 <display-name>Lista de alumnos</display-name>
4 <welcome-file-list>
5 <welcome-file>login.do</welcome-file>
6 </welcome-file-list>
7 <servlet>
8     <servlet-name>dispatcher</servlet-name>
9     <servlet-class>
10         org.springframework.web.servlet.DispatcherServlet
11     </servlet-class>
12     <init-param>
13         <param-name>contextConfigLocation</param-name>
14         <param-value>/WEB-INF/alumno-servlet.xml</param-value>
15     </init-param>
16     <load-on-startup>1</load-on-startup>
17 </servlet>
18 <servlet-mapping>
19     <servlet-name>dispatcher</servlet-name>
20     <url-pattern>/spring-mvc/*</url-pattern>
21 </servlet-mapping>
22 </web-app>
    
```



UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador

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



... continuación **Mi primera aplicación con Spring MVC:**

6º Crear src\main\webapp\WEB-INF\alumno-servlet.xml:

Hay que crear un nuevo fichero alumno-servlet.xml (el fichero xml de configuración del contexto de Spring) acordandose de cambiar base-package="org.alumno.nombreAlumno":

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
  <context:component-scan base-package="org.profesor.joseramon" />
  <mvc:annotation-driven />
```

```
</beans>
```



UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador



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

... continuación **Mi primera aplicación con Spring MVC:**

... continuación 6º Crear src/main/webapp/WEB-INF/alumno-servlet.xml:

El fichero alumno-servlet.xml debería quedar de la siguiente manera:

The screenshot shows an IDE interface. On the left, the Project Explorer displays the project structure for 'joseramon_primer_app_spring_mvc'. The 'src' directory is expanded, showing 'main' (with 'java' and 'resources' subfolders), 'test' (with 'java' and 'resources' subfolders), and 'Libraries'. The 'webapp' directory is also expanded, showing 'WEB-INF'. The file 'alumno-servlet.xml' is selected and highlighted in orange. On the right, the editor shows the content of 'alumno-servlet.xml' with line numbers 1 through 13. The XML code is as follows:

```
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
7       http://www.springframework.org/schema/mvc http://www.springframework.org/s
8       http://www.springframework.org/schema/context http://www.springframework.c
9       <context:component-scan base-package="org.profesor.joseramon" />
10      <mvc:annotation-driven />
11
12 </beans>
13
```




UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador



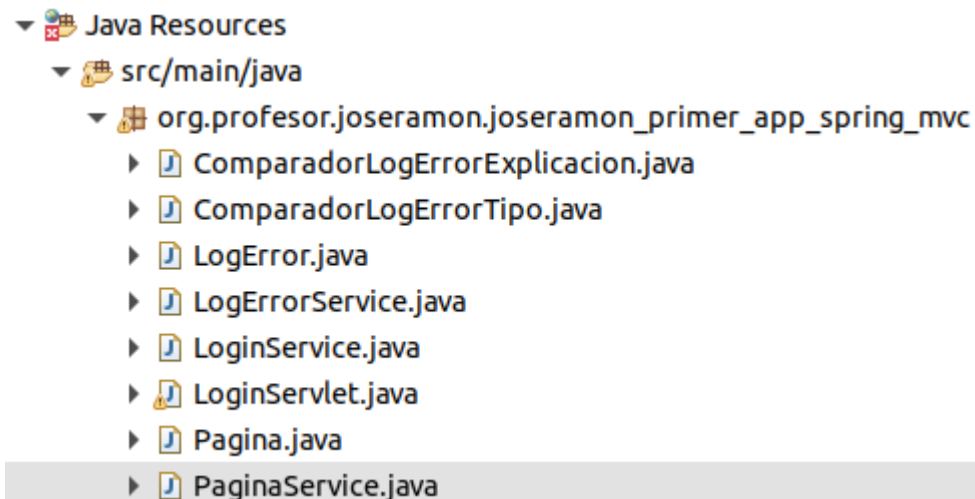
Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com

... continuación **Mi primera aplicación con Spring MVC:**

7º Crear las clases Java y los ficheros JSP para hacer el login inicial:

¿Que clases Java nos hacen falta?

Las vamos a crear inicialmente todas en el mismo paquete (como veremos en las siguientes diapositivas) y más adelante explicaremos como distribuirlas:



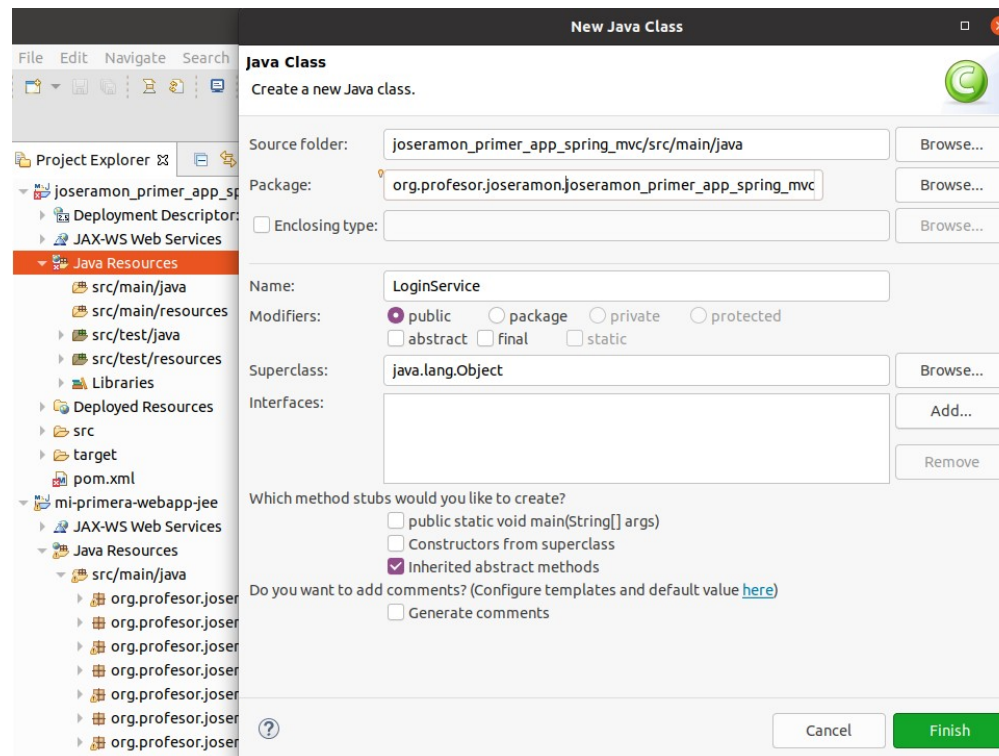


... continuación **Mi primera aplicación con Spring MVC:**

... continuación 7º Crear las clases Java y los ficheros JSP para hacer el login inicial:

Para crear la estructura de paquetes correcta nos colocamos en “Java Resources” y creamos una nueva clase (Java Resources\New \Class) LoginService:

Aunque sea repetitivo es importante que la estructura de paquetes sea **org.alumno.nombreAlumno.nombreAlumno_primer_app_spring_mvc** para que el profesor pueda revisar las prácticas de forma homogenea a todos igual:



UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador

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



... continuación **Mi primera aplicación con Spring MVC:**

... continuación 7º Crear las clases Java y los ficheros JSP para hacer el login inicial:

Ahora podemos **copiar el contenido del fichero LoginService.java** que teníamos en la UD1 pero bajo el nuevo paquete:

```
LoginService.java ✖
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc;
2
3 public class LoginService {
4     public boolean usuarioValido(String usuario,String password) {
5         //En condiciones normales se accedería a una base de datos para
6         //confirmar la contraseña, pero por simplicidad lo validamos
7         //comprobando si son unos valores fijos.
8         //El alumno debe cambiar los valores del usuario y contraseña validos
9         if (usuario.contentEquals("joseramon") && password.contentEquals("miPassword"))
10             return true;
11         return false;
12     }
13 }
```



UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador

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



... continuación **Mi primera aplicación con Spring MVC:**

... continuación 7º Crear las clases Java y los ficheros JSP para hacer el login inicial:

Y debemos hacer lo mismo con las demás clases Java. En el caso de LoginServlet.java haremos una modificación para que vaya a la página “bienvenida.jsp” de momento:

```
LoginServlet.java
8 import javax.servlet.http.HttpServletResponse;
9
10 @WebServlet(urlPatterns = "/login.do")
11 public class LoginServlet extends HttpServlet {
12     LoginService loginServicio = new LoginService();
13     LogErrorService logErroresServicio = new LogErrorService();
14     PaginaService paginaServicio = new PaginaService();
15     static Pagina paginaLogin = new Pagina("Login", "login.do");
16
17     @Override
18     protected void doGet(HttpServletRequest request,
19         HttpServletResponse response) throws IOException, ServletException {
20         paginaServicio.setPagina(paginaLogin);
21         request.getSession().setAttribute("pagina", paginaServicio.getPagina());
22         request.getRequestDispatcher("/WEB-INF/views/login.jsp").forward(request, response);
23     }
24
25     @Override
26     protected void doPost(HttpServletRequest request,
27         HttpServletResponse response) throws IOException, ServletException {
28         String nombre = request.getParameter("nombre");
29         String password = request.getParameter("password");
30         paginaServicio.setPagina(paginaLogin);
31         request.getSession().setAttribute("pagina", paginaServicio.getPagina());
32         request.getSession().setAttribute("numModulosInsertados", 0);
33         if (loginServicio.usuarioValido(nombre, password)) {
34             request.getSession().setAttribute("nombre", nombre);
35             //validación correcta: Redirigir al Servlet de alumno
36             //response.sendRedirect("list-alumno.do");
37             request.setAttribute("nombre", nombre);
38             request.getRequestDispatcher("/WEB-INF/views/bienvenida.jsp").forward(request, response);
39         } else {
40             //validación incorrecta
41             logErroresServicio.addLogError("Login incorrecto", "Login incorrecto de '" + nombre + "'");
42             request.setAttribute("errores", "Usuario '" + nombre + "' o contraseña incorrecta");
43             request.getRequestDispatcher("/WEB-INF/views/login.jsp").forward(request, response);
44         }
45     }
46 }
```


UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador

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



... continuación **Mi primera aplicación con Spring MVC:**

... continuación 7º Crear las clases Java y los ficheros JSP para hacer el login inicial:
Y ahora **copiamos las siguientes páginas .jsp y creamos bienvenida.jsp:**

Project Explorer

joseramon_primer_app_spring_mvc

- Deployment Descriptor: joseramon_primer_app_spring_mvc
- JAX-WS Web Services
- Java Resources
 - src/main/java
 - src/main/resources
 - src/test/java
 - src/test/resources
 - Libraries
- Deployed Resources
- src
 - main
 - java
 - resources
 - webapp
 - WEB-INF
 - jspf
 - footer.jspf
 - header.jspf
 - menuSuperior.jspf
 - views
 - bienvenida.jsp
 - login.jsp
- test
- target
- pom.xml

bienvenida.jsp

```

1 <@ include file="../jspf/header.jspf"%>
2 <@ include file="../jspf/menuSuperior.jspf"%>
3
4 <div class="container">
5   <h1> Bienvenid@ ${nombre}</h1>
6 </div>
7
8 <@ include file="../jspf/footer.jspf"%>
9
10
11

```



UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador

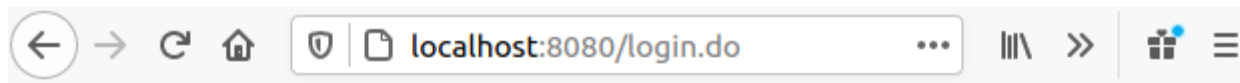


Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com

... continuación **Mi primera aplicación con Spring MVC:**

8º Comprobar que el login funciona:

Aunque la sección de alumnos , modulos y errores no funciona , podemos arrancar la web y probar que hacemos login correctamente y vamos a la página de bienvenida :



[Simarro](#) [Home](#) [Alumnos](#) [Modulos](#) [Errores](#) [DWES](#) [Logout](#)

Introduzca su nombre:

Introduzca su contraseña:

Login



[Simarro](#) [Home](#) [Alumnos](#) [Modulos](#) [Errores](#) [DWES](#) [Logout](#)

Bienvenid@ joseramon

DWES: Desarrollo Web en Entorno Servidor - profesor:



UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador

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



... continuación **Mi primera aplicación con Spring MVC:**

9º Crear nuestro primer controlador Spring MVC : LoginController.java

Realmente todavía estamos trabajando con los Servlets de Java Enterprise Edition 8 (JEE8).

Creemos nuestro primer controlador LoginController.java con el código de la siguiente página en un subpaquete “springmvc”:

Antes de codificar debemos entender que hacen las siguientes **anotaciones Spring MVC** :

@Controller :

Indicamos que se trata de un controlador Spring MVC

@RequestMapping:

Indicamos la ruta del controlador (url accesible desde el navegador)

@ResponseBody:

Indicamos que lo que devolvemos es el cuerpo de la respuesta.



... continuación **Mi primera aplicación con Spring MVC:**

... continuación 9º Crear nuestro primer controlador Spring MVC : LoginController.java

Si comparamos LoginServlet.java que muestra una página HTML con la notación JEE8 y su equivalente LoginController (con la notación Spring MVC) nos damos cuenta que es mucho más sencillo Spring MVC si entendemos las anotaciones :

```
1 package webapp;
2
3 import java.io.IOException;
4
5
6
7
8
9
10
11
12 * Browser sends Http Request to Web Server
13
14
15
16
17
18
19
20 /*Java Platform, Enterprise Edition (Java EE) JEE
21
22 @WebServlet(urlPatterns = "/login.do")
23 public class LoginServlet extends HttpServlet {
24
25     @Override
26     protected void doGet(HttpServletRequest request) {
27         PrintWriter out = response.getWriter();
28         out.println("<html>");
29         out.println("<head>");
30         out.println("<title>Mi primer Servlet</title>");
31         out.println("</head>");
32         out.println("<body>");
33         out.println("Hola Mundo desde mi primer Servlet");
34         out.println("</body>");
35         out.println("</html>");
36     }
37 }
```

```
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.spring_mvc;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.ResponseBody;
6
7 @Controller
8 public class LoginController {
9     @RequestMapping(value="/login")
10     @ResponseBody
11     public String holaMundo() {
12         return "Hola Mundo desde Spring MVC!!";
13     }
14 }
15
```

Markers Properties Servers Data Source Explorer Snippets Console

joseramon_primer_app_spring_mvc [Maven Build] /usr/lib/jvm/java-13-openjdk-amd64/bin/java

INFORMACIÓN: Initializing Spring DispatcherServlet 'dispatcher'

[INFO] Initializing Servlet 'dispatcher'

[INFO] Completed initialization in 600 ms



UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador

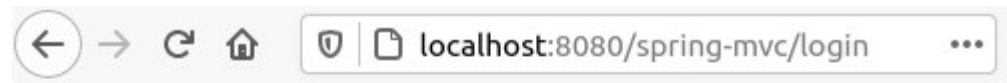
Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com



... continuación **Mi primera aplicación con Spring MVC:**

10º Comprobar que la web nos da la bienvenida:

Solo nos queda **comprobar si funciona accediendo a la ruta spring-mvc/login:**



Hola Mundo desde Spring MVC!!



¿Que pasa si introducimos una url incorrecta?



UD 2: Modelo Vista Controlador

2.- Spring MVC: Controlador



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

... continuación **Mi primera aplicación con Spring MVC:**

... continuación 10º Comprobar que la web nos da la bienvenida:

Si introducimos un url inexistente podemos ver el aviso que se genera en la consola :

The screenshot shows a web browser with the address bar set to `localhost:8080/spring-mvc/loginNoExiste`. The page displays an HTTP 404 status with the message "The requested resource is not available." and the server information "Apache Tomcat/7.0.47".

Below the browser, the IDE's console window is open, showing the following log output:

```
joseramon_primer_app_spring_mvc [Maven Build] /usr/lib/jvm/java-13-openjdk-amd64/bin/java
INFORMACION: Starting ProtocolHandler ["http-bio-8080"]
[WARNING] No mapping for GET /spring-mvc/loginNoExiste
```

A blue arrow points from the 404 error in the browser to the corresponding warning message in the IDE console.



UD 2: Modelo Vista Controlador

2.- Spring MVC: Vista

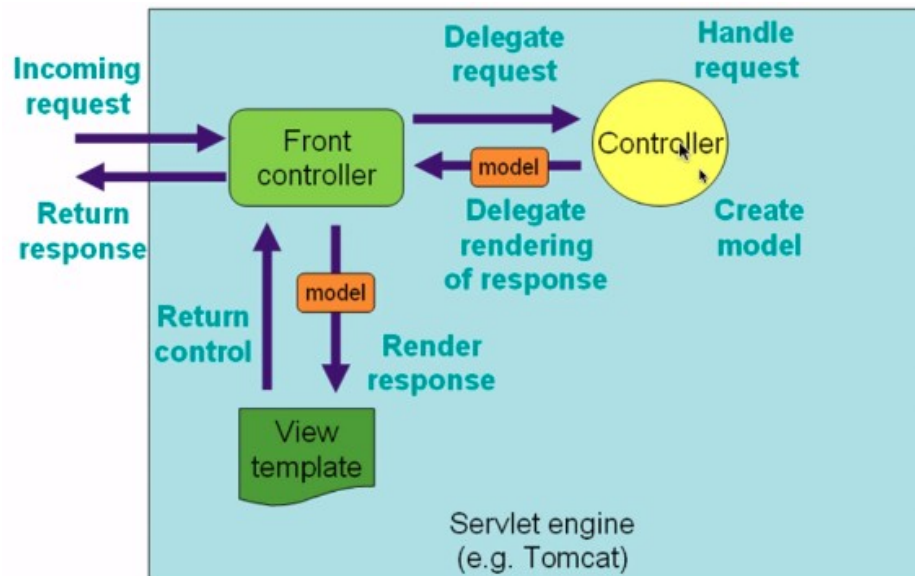
Desarrollo Web en Entorno Servidor - Joseramon.profesor@gmail.com



Recapitulemos lo que hemos visto hasta ahora:

En nuestra primera aplicación hemos configurado nuestro ***DispatcherServlet*** en ***web.xml*** que actúa de Front controller recibiendo todas las peticiones. Para ello hemos tenido que configurar el contexto en ***alumno-servlet.xml***.

Adicionalmente hemos configurado un ***controlador básico (LoginController)*** con las anotaciones ***@Controller*** y ***@RequestMapping***. Por último, hemos hecho que el controlador devuelva un texto al navegador gracias a la anotación ***@ResponseBody***.



Ahora nos queda ver como configurar la vista (ViewResolver) para poder mostrar una página JSP con los datos del modelo (ModelMap). Con esta práctica tendremos un ejemplo completo del patrón FrontController de Spring MVC.



Continuemos configurando nuestra primera web con Spring MVC:

Para mostrar una página JSP en la vista primero que nada vamos a **borrar** la anotación `@ResponseBody`, renombrar el método `holaMundo()` por `mostrarLogin()` e indicarle en el return cual es el nombre de la página JSP queremos mostrar (login):

LoginController.java

```

1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.spring_mvc;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.ResponseBody;
6
7 @Controller
8 public class LoginController {
9     @RequestMapping(value="/login")
10    @ResponseBody
11    public String holaMundo() {
12        return "Hola Mundo desde Spring M
13    }
14 }

```

LoginController.java

```

1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.spring_mvc;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.web.bind.annotation.RequestMapping;
5
6 @Controller
7 public class LoginController {
8     @RequestMapping(value="/login")
9     public String mostrarLogin() {
10         return "login";
11     }
12 }

```



Solo hemos puesto "login", ¿Que debemos configurar para que al decir "login" vaya a "WEB-INF/views/login.jsp"?

UD 2: Modelo Vista Controlador

2.- Spring MVC: Vista

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



Si no configuramos nada más, cuando accedamos a la web nos dará el siguiente error porque al no tener configurado el “View Resolver” se cree que el return es una url, o lo que es lo mismo, piensa que desde login estamos enviando a login y así infinitamente:

HTTP Status 500 - Circular view path [login]: would dispatch back to the current handler URL [/spring-mvc/login] again. Check your ViewResolver setup! (Hint: This may be the result of an unspecified view, due to default view name generation.)

type Exception report

message Circular view path [login]: would dispatch back to the current handler URL [/spring-mvc/login] again. Check your ViewResolver setup! (Hint: This may be the result of an unspecified view, due to default view name generation.)

description The server encountered an internal error that prevented it from fulfilling this request.

exception

javax.servlet.ServletException: Circular view path [login]: would dispatch back to the current handler URL
org.springframework.web.servlet.view.InternalResourceView.prepareForRendering(InternalResourceView

Inspector Consola Depurador Red Editor de estilos Rendimiento

Filtrar las URL

Todos HTML CSS JS XHR Tipografía Imágenes Medios WS Otros

Estado	Método	Dominio	Archivo	Iniciador	Tipo	Transferido	Tamaño	tiempo
500	GET	localhost:8080	login	document	html	3,11 KB	2,91...	30 ms
404	GET	localhost:8080	favicon.ico	FaviconLoader.jsm:179 (img)	html	cacheado	973 B	0 ms

2 solicitudes 3,86 KB / 3,11 KB transferido Finalizado: 117 ms DOMContentLoaded: 42 ms load: 56 ms



UD 2: Modelo Vista Controlador

2.- Spring MVC: Vista



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

Para redirigir a la vista “login.jsp” debemos configurar el View Resolver en `alumno-servlet.xml` que utilizará el Dispatcher Servlet cuando no encuentre `@RequestBody`. Para que solo haga falta escribir “login” y se añada automáticamente el prefijo “/WEB-INF/view/” y el sufijo “.jsp” deberemos añadir el siguiente código en `alumno-servlet.xml`:

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

```
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.
7       http://www.springframework.org/schema/mvc http://www.springframework.org/sc
8       http://www.springframework.org/schema/context http://www.springframework.or
9       <context:component-scan base-package="org.profesor.joseamon" />
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 </beans>
```




UD 2: Modelo Vista Controlador

2.- Spring MVC: Vista



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

El problema se produce porque no encuentran los ficheros de Bootstrap.

```
header.jspf
1 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
2 <!DOCTYPE html>
3 <html>
4 <head>
5 <link href="webjars/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
```

```
footer.jspf
1 <footer class="footer">
2   <p>DWES: Desarrollo Web en Entorno Servidor - profesor: joseramon
3 </footer>
4
5 <script src="webjars/jquery/3.5.1/jquery.min.js"></script>
6 <script src="webjars/popper.js/1.16.0/umd/popper.min.js"></script>
7 <script src="webjars/bootstrap/4.5.2/js/bootstrap.min.js"></script>
8
9 </body>
10
11 </html>
```

Para incorporar correctamente esos ficheros en la siguiente diapositiva introduciremos la ruta en el fichero de configuración del contexto alumno-servlet.xml

UD 2: Modelo Vista Controlador

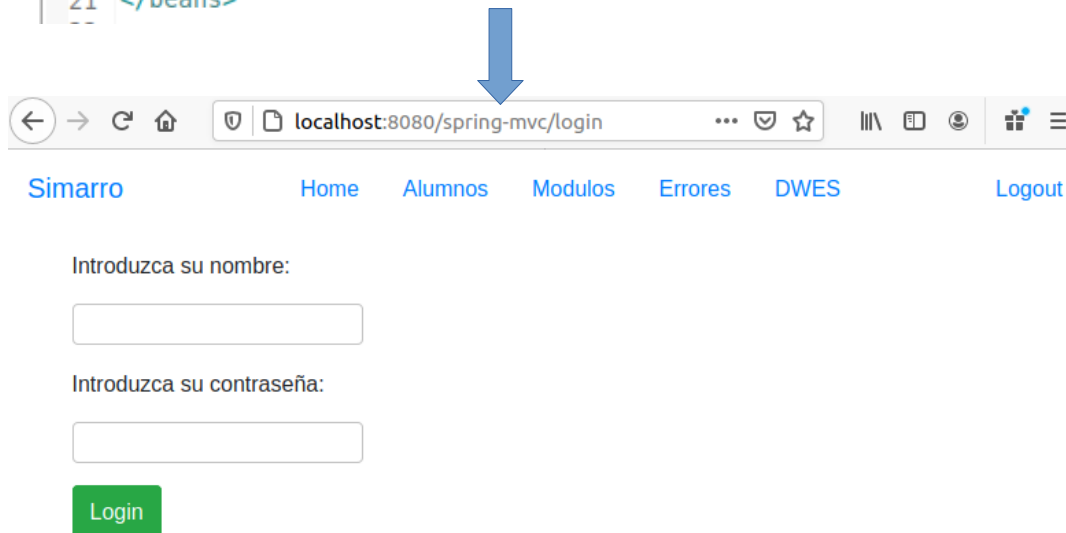
2.- Spring MVC: Vista

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



Si añadimos la línea que incorpora la carpeta webjars en alumno-servlet.xml solucionamos el problema:

```
alumno-servlet.xml
6      xsi:schemaLocation="http://www.springframework.org/sc
7      http://www.springframework.org/schema/mvc http://www.
8      http://www.springframework.org/schema/context http://
9      <context:component-scan base-package="org.profesor.jo
10     <mvc:annotation-driven />
11 <bean
12     class="org.springframework.web.servlet.view.InternalR
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 </beans>
```





UD 2: Modelo Vista Controlador

2.- Spring MVC: Vista

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



Si comprobamos en el navegador el funcionamiento vemos que al pulsar en el botón verde “Login” nos devuelve a la misma página de Login:

localhost:8080/spring-mvc/login

Simarro Home Alumnos Modulos Errores DWES Logout

Introduzca su nombre:

Introduzca su contraseña:

Login

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

Inspector Consola Depurador Red Editor de estilos Rendimiento Memoria

Filtrar las URL

Todos HTML CSS JS XHR Tipografía Imágenes Medios WS Otros

Estado	Método	Dominio	Archivo	Iniciador	Tipo	Transferido	Tam...	0 ms
200	POST	localhost:8080	login	document	html	1,70 KB	1,53 ...	10 ms



¿Por que se comporta así?



El problema viene porque no le hemos especificado que hacer ante un POST, por lo que mostrarLogin() se ejecuta tanto para un GET como para un POST.



¿Como se lo decimos en Spring si no existe el doGet() ni doPost() de JEE8?

```
LoginController.java
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.spring_mvc;
2
3 import org.springframework.stereotype.Controller;
4
5
6 @Controller
7 public class LoginController {
8     @RequestMapping(value="/login")
9     public String mostrarLogin() {
10         return "login";
11     }
12 }
```

UD 2: Modelo Vista Controlador

2.- Spring MVC: Vista

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



Para solucionarlo debemos añadir un nuevo parámetro a `@RequestMapping` indicando el método (method) y un nuevo método “`procesaLogin()`” que atenderá la petición de POST:

```
LoginController.java ✕
1 package org.profesor.josemon.josemon_primer_app_spring_mvc.spring_mvc;
2
3 import org.springframework.stereotype.Controller;
4
5
6 @Controller
7 public class LoginController {
8     @RequestMapping(value="/login",method = RequestMethod.GET)
9     public String mostrarLogin() {
10         return "login";
11     }
12
13
14     @RequestMapping(value="/login",method = RequestMethod.POST)
15     public String procesaLogin() {
16         return "bienvenida";
17     }
18 }
```




Ahora la pregunta es:

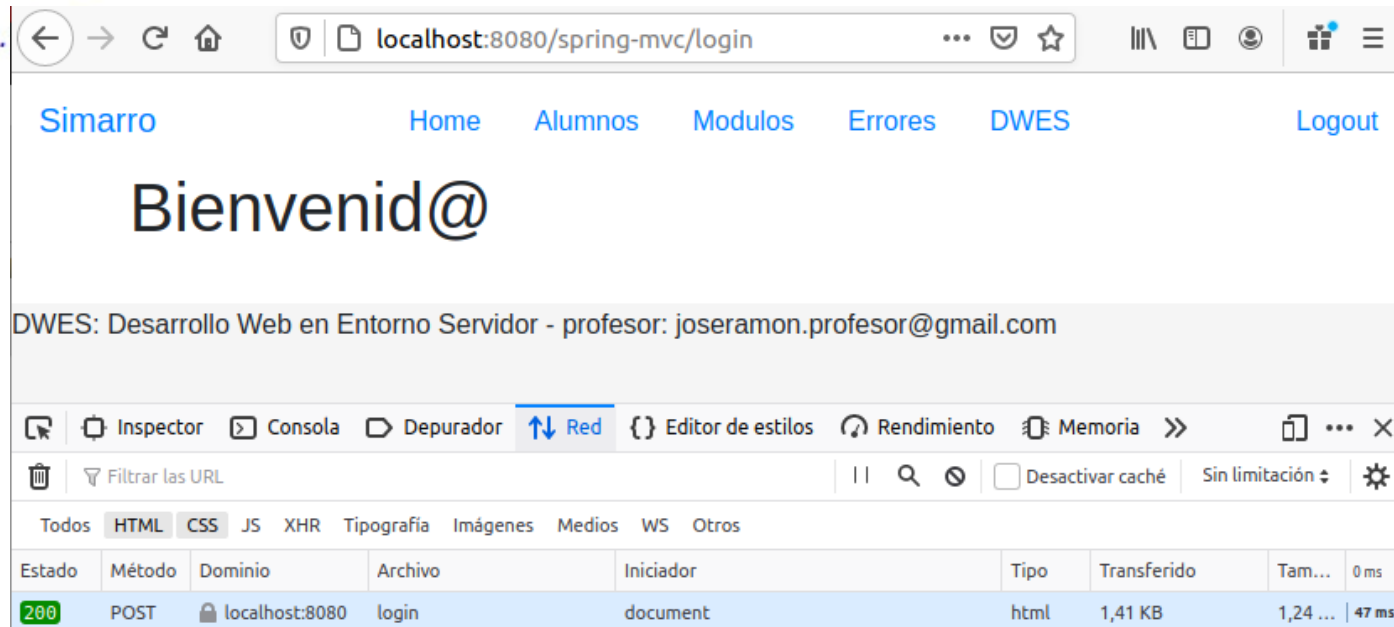
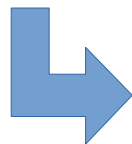


¿ Por que no se muestra el usuario aunque se lo indicamos a bienvenida.jsp?

```

bienvenida.jsp
1 <%@ include file="../../jspf/header.jspf"%>
2 <%@ include file="../../jspf/menuSuperior.jspf"%>
3
4 <div class="container">
5   <h1> Bienvenid@ ${nombre}</h1>
6 </div>
7
8 <%@ include file="../../jspf/footer.

```



UD 2: Modelo Vista Controlador

2.- Spring MVC: Modelo

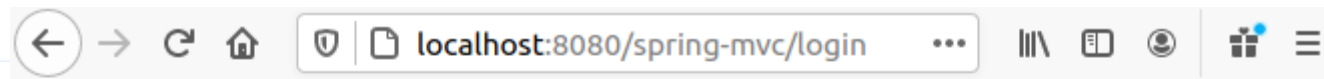
Desarrollo Web en Entorno Servidor - joseramon.profesor@gmail.com



El `request.getParameter()` de JEE8 se debe convertir en un parámetro de entrada (`@RequestParam`) del método en Spring MVC y la forma de configurar un atributo en Spring MVC para mostrarse en el jsp es utilizando una nueva clase `ModelMap` como parámetro de la función y añadiéndole (`model.put()`) los datos que necesitamos.

!!Realiza los cambios!!

```
LoginController.java
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.spring_mvc;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.ui.ModelMap;
5 import org.springframework.web.bind.annotation.RequestMapping;
6 import org.springframework.web.bind.annotation.RequestMethod;
7 import org.springframework.web.bind.annotation.RequestParam;
8
9 @Controller
10 public class LoginController {
11     @RequestMapping(value="/login",method = RequestMethod.GET)
12     public String mostrarLogin() {
13         return "login";
14     }
15
16     @RequestMapping(value="/login",method = RequestMethod.POST)
17     public String procesaLogin(@RequestParam String nombre,
18                               ModelMap model) {
19         model.put("nombre", nombre);
20         return "bienvenida";
21     }
22 }
```



Simarro

Home

Alumnos

Modulos

Errores

DWES

Logout

Bienvenid@ joseramon

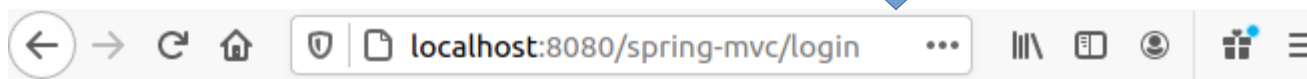


De igual manera , si queremos leer en el controlador más de un valor del formulario de la página jsp, se lo debemos añadir como parámetro al método. Modifica el método y el jsp para leer también el password del usuario y mostrarlo en el jsp:

```
@RequestMapping(value="/login",method = RequestMethod.POST)
public String procesaLogin(@RequestParam String nombre,
    @RequestParam String password,
    ModelMap model) {
    model.put("nombre", nombre);
    model.put("password", password);
    return "bienvenida";
}
```



```
bienvenida.jsp
1 <%@ include file="../../jspf/header.jspf"%>
2 <%@ include file="../../jspf/menuSuperior.jspf"%>
3
4 <div class="container">
5     <h1> Bienvenid@ ${nombre}</h1>
6     Que nadie se entere de que tu password es ${password}
7 </div>
8 <%@ include file="../../jspf/footer.jspf"%>
```

[Simarro](#)[Home](#)[Alumnos](#)[Modulos](#)[Errores](#)[DWES](#)[Logout](#)

Bienvenid@ joseramon

Que nadie se entere de que tu password es miPassword



Ya hemos visto como funciona Spring MVC, pero nos queda una parte importante. El controlador solo debe recibir peticiones, procesarlas y llamar a la vista correspondiente. Para su procesamiento se debe apoyar en los servicios que tiene a su disposición para que el código del controlador sea lo más limpio posible.



¿ Como utilizamos los servicios desde un controlador?

UD 2: Modelo Vista Controlador

2.- Spring MVC: Servicio

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



La primera aproximación es hacerlo al estilo JEE8. Mueve el servicio LoginServicio.java al paquete spring_mvc y modifica el controlador para incorporar el servicio que chequea el login y en función de si es correcto o no vuelve a reenviar a la pagina de login o a la de bienvenida.

Project Explorer
joseramon_primer_app_spring_mvc
JAX-WS Web Services
Java Resources
src/main/java
org.profesor.joseramon.joseramon_primer_app_spring_mvc
org.profesor.joseramon.joseramon_primer_app_spring_mvc.spring_mvc
LoginController.java
LoginService.java
src/main/resources
src/test/java
src/test/resources
Libraries
Deployed Resources
src
main
java
resources
webapp
WEB-INF

LoginController.java

```

7  import org.springframework.web.bind.annotation.RequestParam;
8
9  @Controller
10 public class LoginController {
11     LoginService loginService= new LoginService();
12
13     @RequestMapping(value="/login",method = RequestMethod.GET)
14     public String mostrarLogin() {
15         return "login";
16     }
17
18     @RequestMapping(value="/login",method = RequestMethod.POST)
19     public String procesaLogin(@RequestParam String nombre,
20                               @RequestParam String password,
21                               ModelMap model) {
22         if (!loginService.usuarioValido(nombre, password)) {
23             //usuario inválido, volver a intentar logearse
24             return "login";
25         }
26         // Si llega aquí es porque el usuario era valido
27         model.put("nombre", nombre);
28         model.put("password", password);
29         return "bienvenida";
30     }
31 }

```



Si compruebas el funcionamiento esta todo bien menos una cosa. En caso de introducir las credenciales del usuario incorrectas vuelve al login pero no avisa del error. Modifica el controlador para que se visualice el mensaje de login incorrecto como hacíamos con JEE8:

A screenshot of a web browser window. The address bar shows 'localhost:8080/spring-mvc/login'. The page has a navigation bar with links: Simarro, Home, Alumnos, Modulos, Errores, DWES, and Logout. Below the navigation bar, there is a red error message: 'Usuario 'JoseRa' o contraseña incorrecta'. Underneath the error message, there are two input fields: 'Introduzca su nombre:' and 'Introduzca su contraseña:'. Below the input fields is a green 'Login' button.

← → ↻ 🏠 🔒 localhost:8080/spring-mvc/login ... 📄 📱 👤 ☰

[Simarro](#) [Home](#) [Alumnos](#) [Modulos](#) [Errores](#) [DWES](#) [Logout](#)

Usuario 'JoseRa' o contraseña incorrecta

Introduzca su nombre:

Introduzca su contraseña:

Login

UD 2: Modelo Vista Controlador

2.- Spring MVC: Servicio

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



Ahora vamos a sacarle todo el partido a Spring MVC. No vamos a realizar una explicación teórica, pero es una mala práctica crear el servicio desde el controlador. Es mucho mejor para que el código este desacoplado que el servicio este creado y en el controlador simplemente enlace con dicho servicio.

```
LoginController.java ✖
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.spring_mvc;
2
3 import org.springframework.stereotype.Controller;
4 import org.springframework.ui.ModelMap;
5 import org.springframework.web.bind.annotation.RequestMapping;
6 import org.springframework.web.bind.annotation.RequestMethod;
7 import org.springframework.web.bind.annotation.RequestParam;
8
9 @Controller
10 public class LoginController {
11     LoginService loginService= new LoginService();
12
13     public String mostrarLogin() {}
14
15     @RequestMapping(value="/login",method = RequestMethod.POST)
16     public String procesaLogin(@RequestParam String nombre,
17                               @RequestParam String password,
18                               ModelMap model) {
19         if (!loginService.usuarioValido(nombre, password)) {
```

Para solucionarlo, **Spring** tiene 2 funcionalidades que son la base de su éxito:
La inyección de dependencias y el autoenlace a componentes instanciados.

UD 2: Modelo Vista Controlador

2.- Spring MVC: Servicio

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



Para enlazar LoginService en LoginController debemos hacer que LoginService tenga una instancia valida. Para ello **añadimos la anotación @Service en LoginService.java**, con lo cual Spring en primer lugar instanciará el servicio que después se inyectará en LoginController sin necesidad de que LoginController cree una instancia nueva de la clase LoginService:

```
LoginService.java  ⌕
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.spring_mvc;
2
3 import org.springframework.stereotype.Service;
4
5 @Service
6 public class LoginService {
7     public boolean usuarioValido(String usuario,String password) {
8         //En condiciones normales se accedería a una base de datos para
9         //confirmar la contraseña, pero por simplicidad lo validamos
10        //comprobando si son unos valores fijos.
11        //El alumno debe cambiar los valores del usuario y contraseña validos
12        if (usuario.contentEquals("joseramon") && password.contentEquals("miPassword"))
13            return true;
14        return false;
15    }
16 }
```


UD 2: Modelo Vista Controlador

2.- Spring MVC: Servicio

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



Para que el controlador LoginController puede utilizar este servicio debemos enlazar LoginService en LoginController y para ello **inyectaremos la dependencia LoginService en LoginController mediante la anotación @Autowired**. Fijate que ahora ya no creamos la instancia desde el controlador:

```

LoginController.java
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.spring_mvc;
2
3 import org.springframework.stereotype.Controller;
4
5 @Controller
6 public class LoginController {
7     LoginService loginService= new LoginService();
8
9     @RequestMapping(value="/login",method = RequestMethod.GET)
10    public String mostrarLogin() {
11        return "login";
12    }
13
14    @RequestMapping(value="/login",method = RequestMethod.POST)
15    public String procesaLogin(@RequestParam String nombre,
16                             @RequestParam String password,
17                             ModelMap model) {
18        if (!loginService.usuarioValido(nombre, password)) {
19            //usuario inválido, volver a intentar logearse
20            model.put("errores", "Usuario " + nombre + " o contraseña incorrecta");
21            return "login";
22        }
23        // Si llega aquí es porque el usuario era valido
24        model.put("nombre", nombre);
25        model.put("password", password);
26        return "bienvenida";
27    }
28 }
    
```



```

LoginController.java
1 package org.profesor.joseramon.joseramon_primer_app_spring_mvc.spring_mvc;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Controller;
5 import org.springframework.ui.ModelMap;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RequestMethod;
8 import org.springframework.web.bind.annotation.RequestParam;
9
10 @Controller
11 public class LoginController {
12     @Autowired
13     LoginService loginService;
14
15     @RequestMapping(value="/login",method = RequestMethod.GET)
16    public String mostrarLogin() {
17        return "login";
18    }
19
20    @RequestMapping(value="/login",method = RequestMethod.POST)
21    public String procesaLogin(@RequestParam String nombre,
22                             @RequestParam String password,
23                             ModelMap model) {
24        if (!loginService.usuarioValido(nombre, password)) {
25            //usuario inválido, volver a intentar logearse
26            model.put("errores", "Usuario " + nombre + " o contraseña incorrecta");
27            return "login";
28        }
29        // Si llega aquí es porque el usuario era valido
30        model.put("nombre", nombre);
31        model.put("password", password);
32        return "bienvenida";
33    }
34 }
    
```



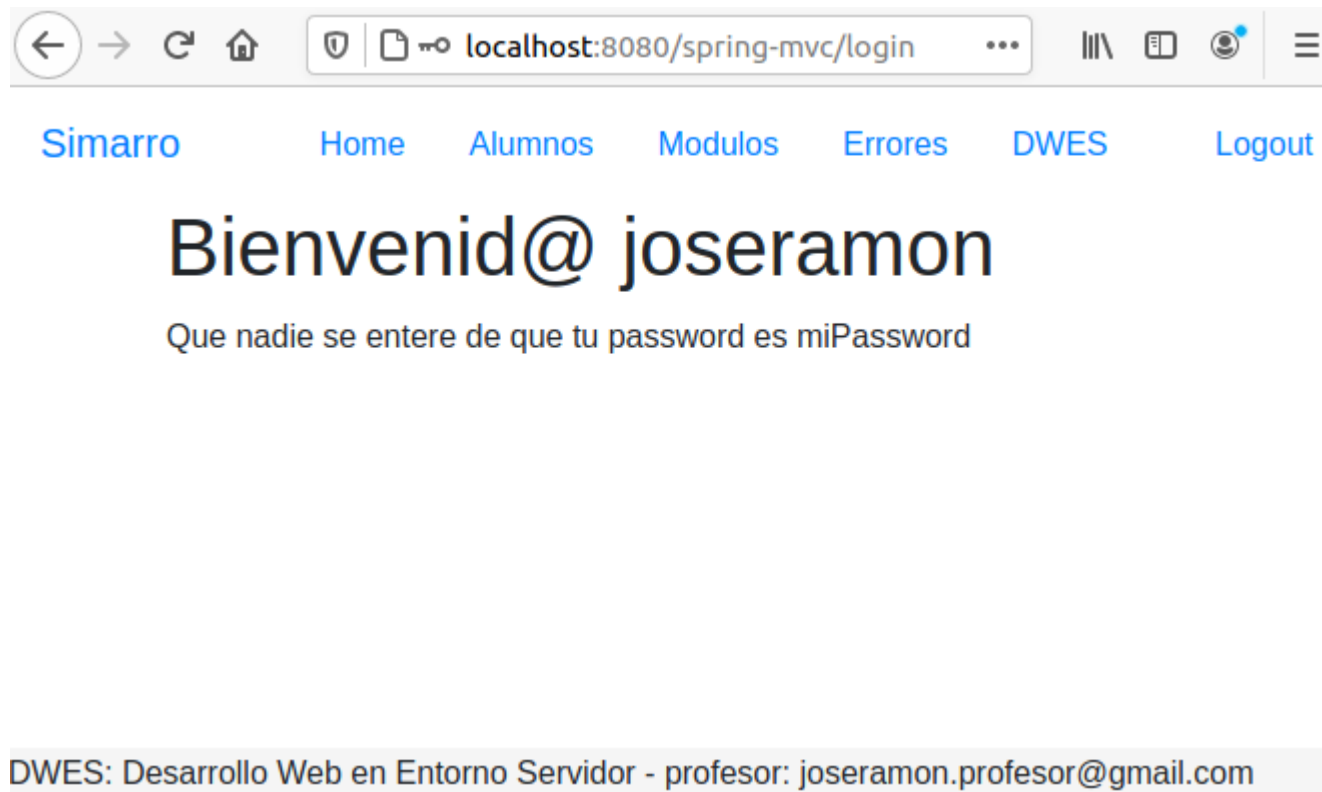
UD 2: Modelo Vista Controlador

2.- Spring MVC: Servicio



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

Realiza los cambios anteriores y comprueba que la web sigue funcionando. Puede que necesites parar y relanzar la web:





UD 2: Modelo Vista Controlador

2.- Spring MVC: Servicio

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



EJERCICIO:

Sigue todos los pasos de los PDF y 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_practica1_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.