¿Qué es un controlador en Spring MVC?

- Componentes que proporcionan acceso al funcionamiento de la aplicación.
  - ✓ Procesan la lógica de la aplicación (se apoyan de componentes de la capa de servicio).
  - ✓ Reciben los datos de entrada del usuario (ej. datos de un formulario HTML).
  - ✓ Crean el modelo (datos de la aplicación) y regresan el nombre de la vista al DispatcherServlet.
- Anotaciones específicas de Spring MVC.
  - ✓ @Controller, @RequestMapping, @RequestParam, @ModelAttribute, etc.

HomeController.java

```
package net.itinajero.app.controller;
// imports

@Controller
public class HomeController {

    @RequestMapping(value="/detail")
    public String mostrarDetalle(Model model) {
     String tituloPelicula = "Rapido y Furioso 8";
     model.addAttribute("titulo", tituloPelicula);
     return "detalle";
    }

}
```

Configuración del DispatcherServlet

```
<context:component-scan
base-package="net.itinajero.app.controller" />
```

detalle.jsp

```
<html>
  <body>
    <h1> TITULO DE LA PELICULA ${ titulo } </h1>
  </body>
</html>
```

Podemos usar Expression Languaje y JSTL en los archivos JSP para mostrar los objetos del modelo que son compartidos por el controller.

Configuración de JSTL

- La librería JSTL es un componente dentro de la especificación de Java EE.
- Conjunto de etiquetas simples, que agregan funcionalidad a las páginas JSP.
  - ✓ Loops
  - ✓ Condicionales
  - ✓ Entrada / Salida
  - ✓ Etc.
- En una aplicación de Spring MVC, típicamente se utiliza solo para desplegar en las vistas (archivos JSP) los datos del modelo.
- Para agregar el soporte de JSTL utilizando Maven (pom.xml)

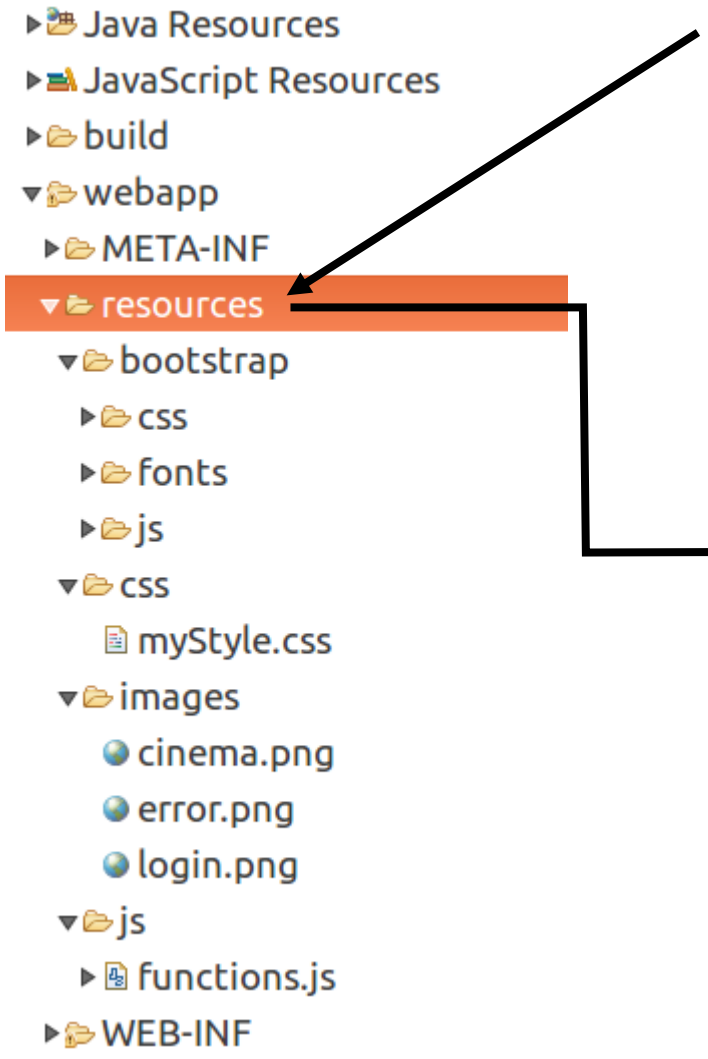
```
<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

- Para importar JSTL en los JSP, se agregaría:

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
```

Core: loops, condicionales
Fmt: Para dar formato, i18n.

Configuración de recursos estáticos (JS, CSS, Images)



1. Crear carpeta "resources" donde se almacenarán los recursos estáticos.
   - bootstrap: archivos js, css y fonts del framework front-end bootstrap.
   - css: otros archivos CSS
   - images: archivos de imagen (png, jpg, etc).
   - js: nuestros archivos con funciones Javascript.
2. Agregar la siguiente configuración al archivo XML del DispatcherServlet.

```
<mvc:resources mapping="/resources/**" location="/resources/" />
```

3. Incluir el tag library de Spring en el archivo JSP.

```
<%@taglib uri="http://www.springframework.org/tags"
prefix="spring"%>
```

4. Agregar una variable al modelo con la URL relativa a resources.

```
<spring:url value="/resources" var="urlPublic" />
```

5. Utilizar la variable antes creada para acceder a los archivos estáticos.

```
<link href="${urlPublic}/css/myStyle.css" rel="stylesheet">
<img src="${urlPublic}/images/cinema.png">
```