

**Tipo** : Guía de laboratorio  
**Capítulo** : Spring MVC  
**Duración** : 40 minutos

---

## I. OBJETIVO

Conocer las diferentes formas que tiene el framework de trabajar con el Controlador (mediante el uso de las clases Model, ModelAndView, etc.) y la configuración estándar de las vistas.

## II. REQUISITOS

Los siguientes elementos de software son necesarios para la realización del laboratorio.

- NetBeans 11
- JDK 11

## III. EJECUCIÓN DEL LABORATORIO

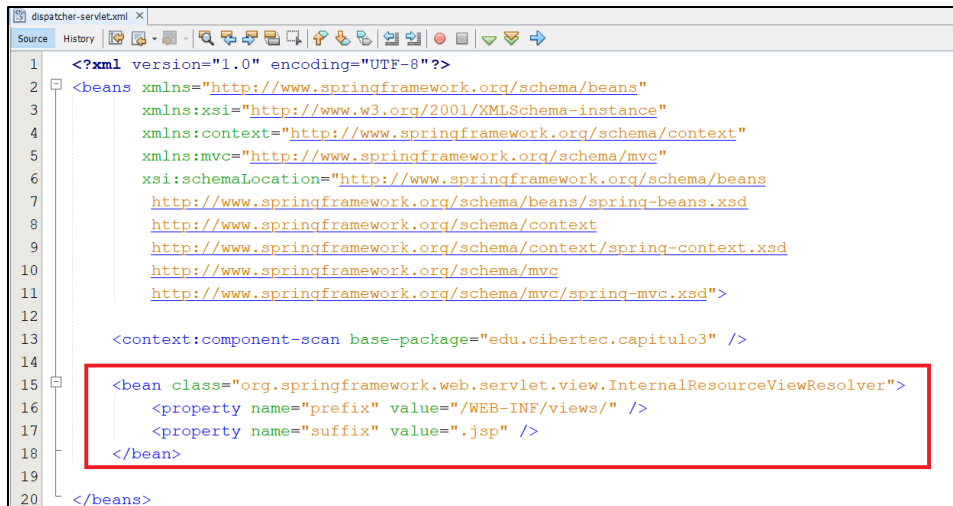
### Ejercicio:

Modifica el proyecto para estandarizar la forma de invocar a las pantallas y comprueba las diferentes formas que se tienen para capturar y pasar parámetros desde/hacia las pantallas.

1. Spring permite la estandarización de llamados a las pantallas en el framework, de tal manera que puedes indicar el tipo de resultado a presentar, la carpeta donde estarán alojadas y la extensión que usarán. Para poder realizar esto, debes utilizar un ViewResolver.

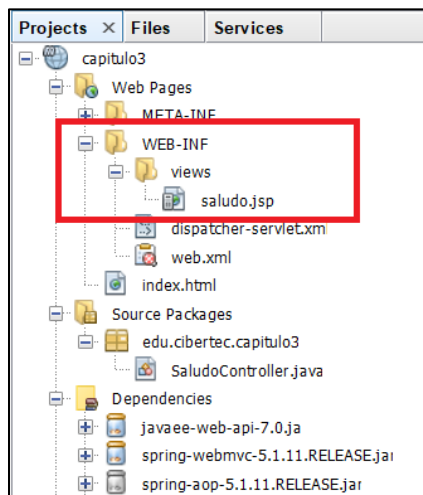
La aplicación establece que los archivos a mostrar están alojados dentro de la carpeta "WEB-INF/views" (esto se hace para proteger a las pantallas para que no puedan ser accedidas desde URLs externos) y que la extensión de las pantallas es "\*.jsp".

Para inscribir el ViewResolver, modifica el archivo configurador del Servlet añadiendo las siguientes líneas.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:context="http://www.springframework.org/schema/context"
5       xmlns:mvc="http://www.springframework.org/schema/mvc"
6       xsi:schemaLocation="http://www.springframework.org/schema/beans
7                           http://www.springframework.org/schema/beans/spring-beans.xsd
8                           http://www.springframework.org/schema/context
9                           http://www.springframework.org/schema/context/spring-context.xsd
10                          http://www.springframework.org/schema/mvc
11                          http://www.springframework.org/schema/mvc/spring-mvc.xsd">
12
13     <context:component-scan base-package="edu.cibertec.capitulo3" />
14
15     <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">
16         <property name="prefix" value="/WEB-INF/views/" />
17         <property name="suffix" value=".jsp" />
18     </bean>
19
20 </beans>
```

2. Para que la aplicación siga funcionando, mueve el archivo “saludo.jsp” a la carpeta “WEB-INF/views” (que aún no existe) para que quede de la siguiente manera.

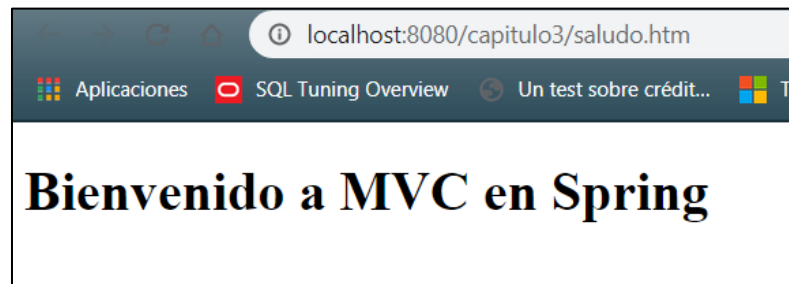


3. El último cambio será en el Controller, donde debes quitar la extensión que tenía el archivo para que ahora sea completado directamente por el framework. El Controller queda de la siguiente manera.

```
@Controller
public class SaludoController {

    @RequestMapping("saludo")
    public String saludar() {
        return "saludo";
    }
}
```

4. Ejecuta el proyecto y comprueba que siga funcionando sin problemas.



5. Ahora observa cómo se puede pasar un parámetro a la página que vas a invocar. Existen muchas formas de poder lograr el envío de información, empieza viendo el uso de la clase Model.

Modifica el controller de la siguiente manera para añadir un mensaje de bienvenida en la página de saludo.

```
@RequestMapping("saludo")
public String saludar(Model modelo) {
    modelo.addAttribute("mensaje", "Bienvenido desde el controlador");
    return "saludo";
}
```

6. Modifica la página de saludo para que imprima el objeto “mensaje” que se le está enviando.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Primera Aplicación Spring MVC</title>
</head>
<body>
<h1>${mensaje}</h1>
</body>
</html>
```

7. Ejecuta el proyecto y obtén la siguiente respuesta.



8. Emplea otra clase que no solo contiene el modelo, sino también la navegación de las páginas. La clase es ModelAndView y la puedes implementar de la siguiente manera.

```
public class SaludoController {  
  
    @RequestMapping("saludo")  
    public ModelAndView saludar() {  
        ModelAndView mv = new ModelAndView();  
        mv.addObject("mensaje", "Bienvenido desde el controlador");  
        mv.setViewName("saludo");  
        return mv;  
    }  
}
```

9. Ejecuta la aplicación y sigue obteniendo la respuesta anterior sin ningún cambio.



10. También se puede implementar el método con menos líneas gracias a la diversidad de firmas con que cuenta la clase ModelAndView; por ejemplo, puedes implementarlo de la siguiente manera.

```
@RequestMapping("saludo")  
public ModelAndView saludar() {  
    return new ModelAndView("saludo", "mensaje", "Bienvenido desde el controlador");  
}
```

#### IV. EVALUACIÓN

Responde la siguiente pregunta.

1. ¿Cuál es la diferencia de las extensiones que se inscriben en el ViewResolver y en el DispatcherServlet?

La diferencia es que la extensión del DispatcherServlet indica que se interceptará cualquier llamado (request) que lleve esa extensión para ser manejada por Spring. Mientras que la extensión del ViewResolver es para determinar los tipos de archivos que serán invocados como vista.