Dispatcher

¡Comencemos!

¿Qué aprenderás?

- Crear e implementar el Dispatcher (Despachador) en el proyecto holasmundospringmvc mediante el archivo web.xml.

Introducción

El dispatcher, es parte del Front Controller como se ha descrito anteriormente en este documento. Este, es el punto de entrada para todas las peticiones del cliente. En este apartado, explicaremos cómo implementar el dispatcher en nuestro archivo web.xml y probar el proyecto holamundospringmvc que hemos estado desarrollando como ejemplo.

¡Vamos con todo!

Dispatcher

En el apartado anterior, aprendimos como crear un proyecto Spring MVC desde cero, utilizando las mejores herramientas para que el proceso sea simple e intuitivo. El Dispatcher, es un servlet que nos proporciona Spring MVC, el cual tiene la responsabilidad de controlar todas las peticiones del cliente al o los controladores que debemos desarrollar para que nuestra aplicación funcione. Este servlet, es considerado parte del Front Controller que define la arquitectura de Spring. Este funciona de la siguiente manera:

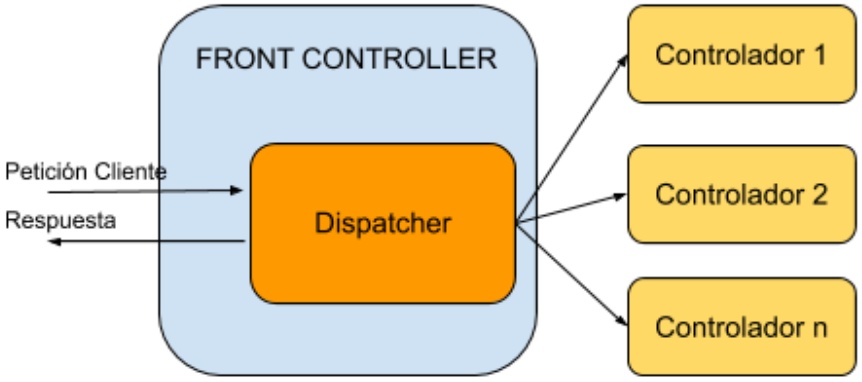

Imagen 1. Diagrama de Dispatcher.
Fuente: Desafío Latam.

En la imagen 1, se muestra como funciona el dispatcher de la aplicación. Este es el encargado de gestionar y despachar las peticiones de los clientes.

Por lo que si intentamos acceder a la URL, el dispatcher buscará en los controladores el método welcomeUser y cuando sea encontrado lo invocará, y si no lo encuentra nos devolverá un error. Debido a esto es la importancia de comprender este capítulo, para poder procesar las url que solicita un navegador y responder invocando la correspondiente acción.

Para que lo anterior funcione de forma práctica, vamos a utilizar un archivo llamado web.xml, el cual se debe implementar en todos los proyectos de esta índole. Este archivo, es el primero en cargarse en el servidor de aplicaciones, ya que es el que posee la configuración inicial de carga del proyecto.

Para configurar entonces, el dispatcher de nuestro proyecto, debemos hacer lo siguiente:

1. En src/main/webapp/ crear la carpeta WEB-INF dentro del directorio src/main/webapp de nuestro proyecto.

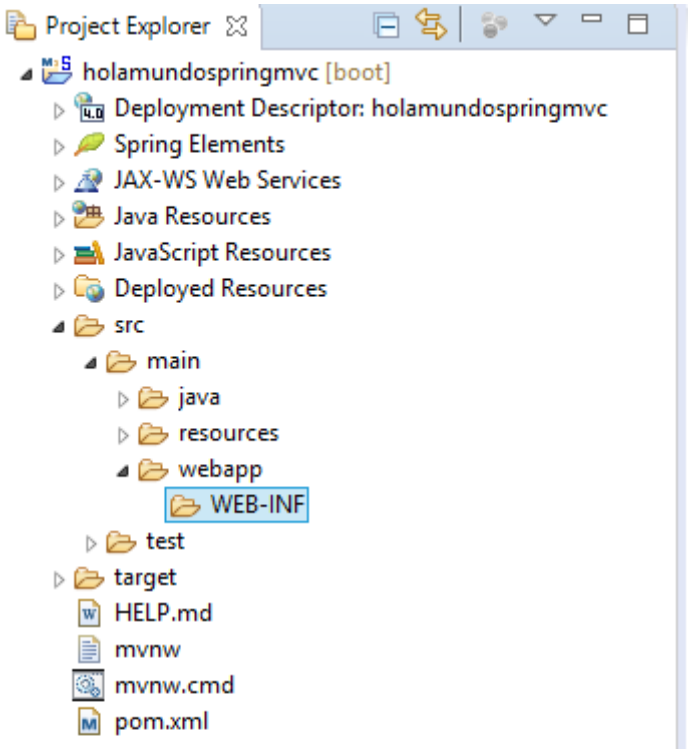

Imagen 2. Carpeta WEB.INF.
Fuente: Desafío Latam.

2. Crear el archivo web.xml, dentro de la carpeta WEB-INF, con el siguiente código:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app metadata-complete="true"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://Java.sun.com/xml/ns/javaee"
xmlns:web="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
id="WebApp_ID"
version="3.0">
<display-name>holamundospringmvc</display-name>
 <servlet>
       <servlet-name>spring</servlet-name>
           <servlet-class>
               org.springframework.web.servlet.DispatcherServlet
           </servlet-class>
       <load-on-startup>1</load-on-startup>
   </servlet>
   <servlet-mapping>
       <servlet-name>spring</servlet-name>
       <url-pattern>/*</url-pattern>
```

```
    </servlet-mapping>
</web-app>
```

- Entre los tag <servlet></servlet>, indica la sección donde se declara que servlet, el cual se usará para gestionar las peticiones del cliente.

- org.springframework.web.servlet.DispatcherServletes el dispatcher que Spring MVC nos provee.

3. Volvemos a arrancar el proyecto, con la configuración que ya hemos guardado. De esta forma, comprobamos que todo está bien configurado y en orden.

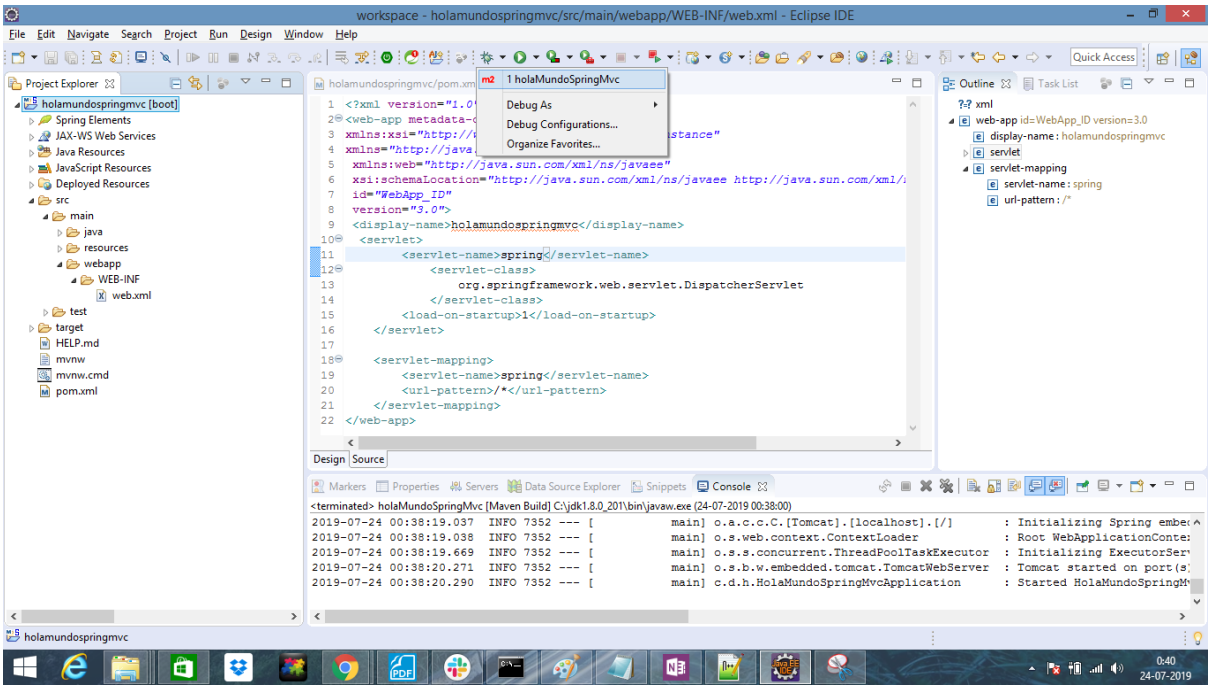

Imagen 3. Iniciando la aplicación.
Fuente: Desafío Latam.

- Finalmente, debemos hacer click en el círculo verde con el triángulo dentro y luego en holaMundoSpringMvc, como se muestra en la imagen anterior.