

Agregar Spring MVC a una aplicación web

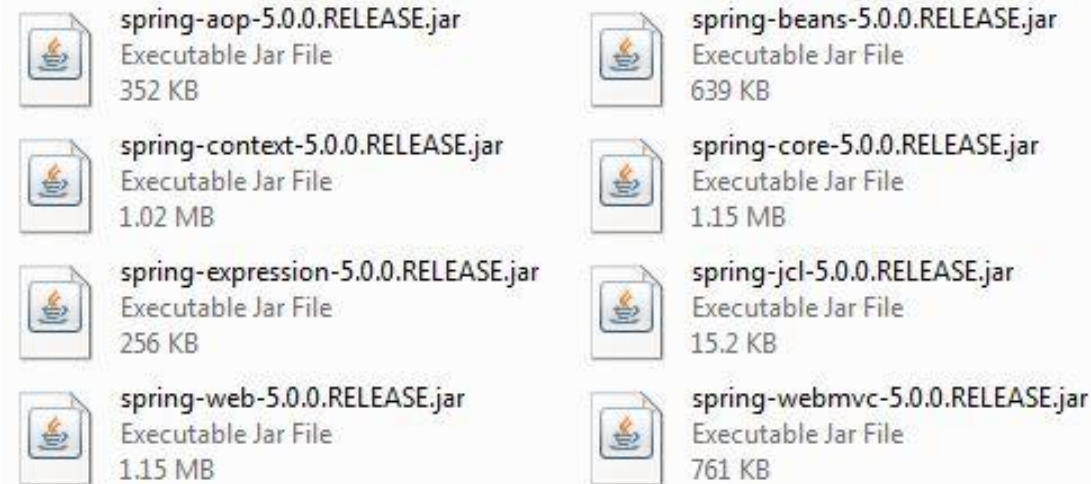
➤ La forma más común de agregar Spring MVC es utilizar Maven.

- ✓ Abrir el archivo pom.xml en STS.
- ✓ Dentro del tag <dependencies> agregar la siguiente dependencia:

```
<dependencies>

  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.0.0.RELEASE</version>
  </dependency>

</dependencies>
```



Última versión al día de hoy : 5 . x . x . RELEASE

<http://projects.spring.io/spring-framework/>

Recomendación: Cambia esta versión por la última al día cuando estes viendo esta lección.

**Librerías (JARs)
descargados.**

Configuración del DispatcherServlet (1)

- El DispatcherServlet es un **Servlet** (extends HttpServlet).
- Debe ser declarado en el archivo **web.xml**
- Configuración estándar de Java EE Servlet.

```
<servlet>
  <servlet-name>springmvc</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>springmvc</servlet-name>
  <url-pattern>*.html</url-pattern>
</servlet-mapping>
```

- Al iniciar el DispatcherServlet Spring MVC buscará el archivo [servlet-name]-servlet.xml en el directorio WEB-INF. Este archivo contendrá todos los BEANS y configuración de Spring MVC. Para este ejemplo intentará buscar el archivo /WEB-INF/springmvc-servlet.xml

Configuración del DispatcherServlet (2)

➤ Se puede cambiar la ubicación del archivo de configuración de Spring MVC.

Archivo web.xml

```
<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/spring/dispatcher-servlet.xml</param-value>
  </init-param>
  <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
  <servlet-name>dispatcher</servlet-name>
  <url-pattern>*.html</url-pattern>
</servlet-mapping>
```

Configuración de Spring MVC

- La configuración debe de ir en el archivo definido en el DispatcherServlet [web.xml]
- Como mínimo debe tener la siguiente configuración.
 - ✓ Agregar 3 namespaces [beans, context, mvc]
 - ✓ Indicarle al Spring los paquetes que deberá escanear automáticamente para encontrar BEANS con las anotaciones (@Component, @Controller, @Service, @Repository)
 - ✓ Habilitar el soporte de anotaciones que son específicas de Spring MVC (@Controller, @RequestMapping, @RequestBody, @ResponseBody, etc).
 - ✓ Definir un ViewResolver (indicarle a Spring MVC donde buscará las vistas [JSP])

Archivo /WEB-INF/springmvc-servlet.xml

```
<beans [ namespaces ] >
  <context:component-scan base-package="net.itinajero.app.controller" />
  <mvc:annotation-driven />
  <bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix">
      <value>/WEB-INF/views/</value>
    </property>
    <property name="suffix">
      <value>.jsp</value>
    </property>
  </bean>
</beans>
```

Estructura de un Controller

- Un Controller en Spring MVC es una clase normal a la cual se le agrega la anotación `@Controller` a nivel de la clase.
- Puede tener varios métodos con la anotación `@RequestMapping` (Action Controller).
- Los métodos pueden tener cualquier nombre y deben regresar un String (VIEW).
- Los métodos son ejecutados al ser invocados por medio de la URL especificada en el parámetro **value** de `@RequestMapping`. El parámetro **method** es opcional. Si no se especifica, responderá a cualquier tipo de petición (GET, POST, DELETE, PUT, etc).

```
@Controller
public class HomeController {

    @RequestMapping(value = "/home", method=RequestMethod.GET)
    public String goHome() {
        // Mi lógica de negocio
        return "home";
    }
}
```

PETICIÓN TIPO GET

localhost:8080/holaMundo/home.html

El DispatcherServlet
buscará una vista llamada
WEB-INF/views/**home.jsp**

WebApplicationContext (1)

➤ ¿Qué es un Context en Java?

- ✓ Es el entorno de ejecución compuesto por varios componentes (**variables de entorno, variables de instancia, estado de las clases, alcance de objetos, sesiones**) para una aplicación en particular.


- Ejemplos:

- En una aplicación Java Web EE existe el **ServletContext** que representa el entorno de ejecución para una aplicación web dentro del contenedor (Apache Tomcat).
- En una aplicación desarrollada con Spring existe el **ApplicationContext** que representa el núcleo de una aplicación (Beans) dentro del contenedor de Spring.

➤ ¿Qué es un WebApplicationContext?

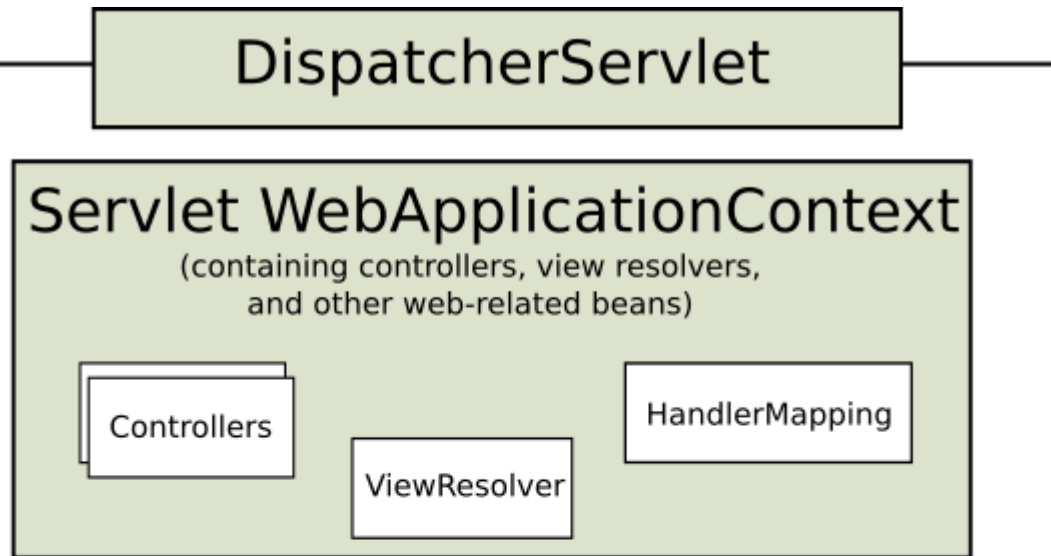
- ✓ En Spring MVC es un tipo de Context basado en Servlets que es creado en base a la configuración del Dispatcher Servlet (pueden ser varios) ubicada en el archivo web.xml de la aplicación web.
- ✓ Cada Dispatcher Servlet tiene su propio Context y es inicializado a partir del archivo

<servlet-name>-servlet.xml = springmvc-servlet.xml



```
<servlet>
  <servlet-name>springmvc</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <load-on-startup>1</load-on-startup>
</servlet>
```

WebApplicationContext (2)



Un `WebApplicationContext` por lo general solo tiene la declaración de componentes web (beans)

- `Controllers`
- `View Resolvers`
- `HandlerMapping`