Universidad Andrés Bello®
Conectar · Innovar · Liderar

DESARROLLO DE APLICACIONES JEE CON SPRING FRAMEWORK (V2)

El Framework Spring MVC

Características del Framework Spring MVC

¿Qué es Spring Framework?

Spring Framework es una herramienta que permite desarrollar aplicaciones web a travésdel patrón de diseño Modelo-Vista-Controlador (MVC). Está diseñado en torno a un DispatcherServlet que envía solicitudes a los controladores, con mapas de control configurables, resolución de vista, resolución de localización, resolución de temas y soporte para la carga de archivos. El controlador predeterminado se basa en @Controller y @RequestMapping, que ofrecen una amplia gama de métodos de control flexibles. El mecanismo @Controller también permite crear aplicaciones y páginas Web a través de anotaciones en la @PathVariable.

En Spring Web MVC se puede utilizar cualquier objeto como un comando o como un formulario, de manera que no se necesita implementar una interfaz específica o una clase. El enlace a los datos es muy flexible; por ejemplo, trata a los errores de tipo como errores de validación y no como errores del sistema, de manera que puedan ser evaluados por la aplicación. La resolución de la vista es extremadamente flexible.

El controlador es normalmente el responsable de preparar un modelo Map con los datosy seleccionar un nombre de vista, pero también puede escribir directamente en lasecuencia de respuesta y completar la solicitud. La resolución del nombre de la vista es configurable a través de la extensión del archivo, usando los nombres de los beans o incluso mediante la implementación personalizada del ViewResolver. El modelo es un mapa de la interfaz, lo que permite obtener una abstracción completa de la de vista. Se
puede integrar la vista directamente con plantillas basadas en las tecnologías de representación tales como JSP, Velocity y Freemarker o directamente generar XML, JSON, Atom, y muchos otros tipos de contenidos. El modelo Map es simplemente transformadoen el formato adecuado, como los atributos de la petición JSP o un modelo de plantilla Velocity.

¿Qué características presenta Spring Framework?

El módulo de Spring Web incluye muchas características únicas de soporte Web:

- **Separación de los roles:** Cada rol puede ser cumplido por un objeto especializado.

- **Configuración:** La configuración es potente y directa, tanto del framework y las clases de la aplicación como del JavaBeans. Esto facilita la consulta a través de contextos, como el de los controladores Web para los objetos y los validadores.

- **Adaptabilidad, no injerencia y flexibilidad:** Es posible definir cualquier método controlador que se necesite, usando una de las anotaciones de parámetros (@RequestParam, @RequestHeader, @PathVariable, etc.) para un escenario dado.

- **Reutilización de código:** Utilización de los objetos existentes como comandos o formularios, en lugar de duplicarlos para extender una clase particular del framework.

- **Enlaces y validación personalizados:** Trata los errores de tipo como errores de validación a nivel de aplicación que mantienen el valor que lo produjo.

- **Asignación de control y resolución de la vista personalizable:** Estrategias para la asignación de control y la resolución de las vistas que van desde la configuración en base a URL sencillas, hasta sofisticadas estrategias de resolución de propósitos.

- **Transferencia de modelo flexible:** Transferencia de modelo con un nombre/valor Map, el cual es integrado fácilmente con cualquier tecnología de vistas.

- **Versatilidad:** Obtención de la configuración regional, de la zona horaria y del tema; posee soporte para JSP con o sin etiquetas de bibliotecas, soporte para JSTL, soporte para Velocity sin necesidad de puentes.

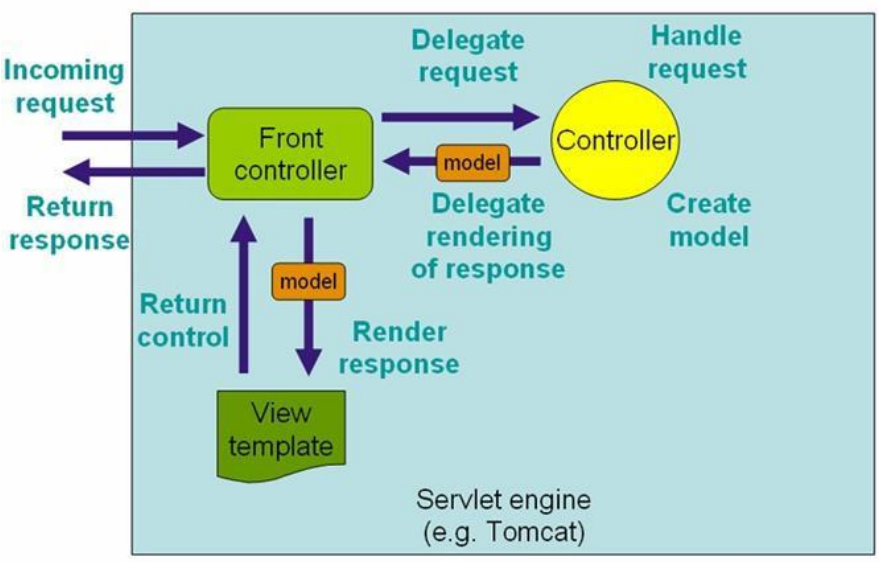
- **Biblioteca:** Una biblioteca de etiquetas JSP que proporciona soporte para funciones como el enlace de datos y temas. Además, hace que la escritura en las páginas JSP sea mucho más fácil.

- **Beans:** son elementos cuyo ciclo de vida está en el ámbito de la actual HTTPRequest o HTTPSesion. Esto no es una característica específica de Spring MVC, sino más bien del recipiente WebApplicationContext que utiliza Spring MVC.

El DispatcherServlet

Spring Web framework MVC es impulsado por una solicitud, y está diseñado en torno a un Servlet central que envía solicitudes a los controladores y ofrece funcionalidades que facilitan el desarrollo de aplicaciones web, sin embargo, el DispatcherServlet de Spring nose limita a eso, está completamente integrado con el contenedor IoC, y como tal le permite usar todas las otras características proporcionadas por Spring.

El flujo de procesamiento de solicitudes de Spring Web MVC DispatcherServlet se ilustra en el siguiente diagrama. El DispatcherServlet es una expresión del patrón de diseño Front Controller.



El DispatcherServlet es un Servlet real (hereda de la clase HttpServlet), y como tal se declara en el archivo web.xml de la aplicación web. Es necesario seleccionar las peticionesque el DispatcherServlet vaya a manejar mediante la asignación de una dirección URL enel archivo web.xml.

A continuación, se muestra un ejemplo de la declaración del DispatcherServlet:

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

    <servlet-mapping>
        <servlet-name>example</servlet-name>
        <url-pattern>/example/*</url-pattern>
    </servlet-mapping>

</web-app>
```

En el ejemplo anterior, todas las peticiones que empiezan por /example son manejadas por la instancia del DispatcherServlet llamada example.

A partir de la versión Servlet 3.0 también se puede configurar el contenedor de programación de Servlets.

A continuación, se muestra el código equivalente al ejemplo anterior:

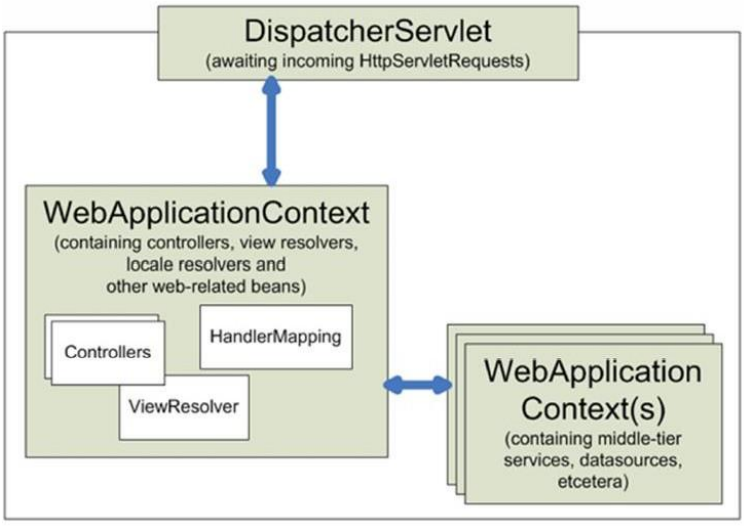
```
public class MyWebApplicationInitializer implements
WebApplicationInitializer {

    @Override
    public void onStartup(ServletContext container) {
        ServletRegistration.Dynamic registration =
container.addServlet("dispatcher", new DispatcherServlet());
        registration.setLoadOnStartup(1);
        registration.addMapping("/example/*");
    }

}
```

WebApplicationInitializer es una interfaz proporcionada por Spring MVC que detecta la configuración basada en código y automáticamente la utiliza para inicializar cualquier contenedor del Servlet. Una implementación de la clase abstracta llamada AbstractDispatcherServletInitializer hace que sea aún más fácil de registrar el DispatcherServlet.

Cada DispatcherServlet tiene su propio WebApplicationContext, el cual hereda todos los beans que ya están definidos en la raíz del WebApplicationContext. Estos beans se pueden sobrescribir, además de crear nuevos beans en una instancia de un servlet determinado.



En la inicialización de un DispatcherServlet, Spring MVC busca un archivo llamado [servletname]- servlet.xml en el directorio WEB-INF de la aplicación web y crea los beans definidos allí, ignorando las definiciones de cualquier bean definido con el mismo nombreen ámbito global.

A continuación se muestra la configuración de un DispatcherServlet en el archivo web.xml:

```
<web-app>
    <servlet>
        <servlet-name>golfing</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>golfing</servlet-name>
        <url-pattern>/golfing/*</url-pattern>
    </servlet-mapping>
</web-app>
```

Con la configuración del servlet anterior, la aplicación tendrá que tener un archivo llamado /WEB-INF/golfing-servlet.xml; este archivo contendrá todos los componentes específicos (beans) de Spring Web MVC. WebApplicationContext es una extensión de ApplicationContext que tiene algunas características adicionales necesarias para las aplicaciones web.

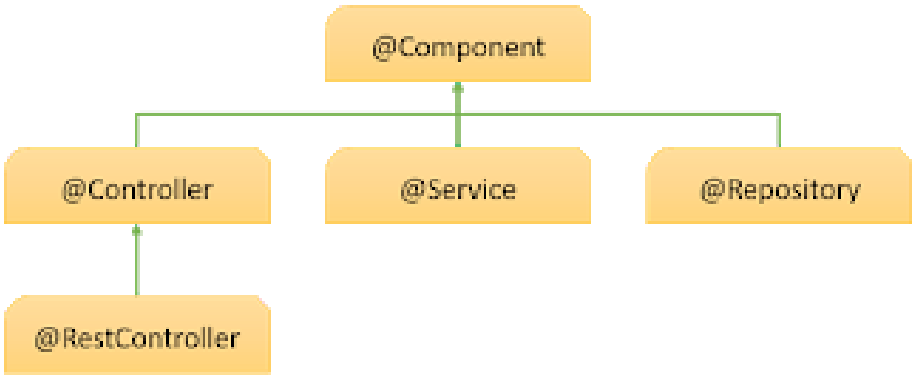
Se diferencia de ApplicationContext en que es capaz de resolver temasy que conoce con que Servlet está asociado, ya que tiene un enlace al ServletContext. WebApplicationContext está unido a un ServletContext, y mediante el uso de métodos estáticos en la clase RequestContextUtils es posible acceder si es necesario al WebApplicationContext.

El concepto y uso de anotaciones

En los diversos desarrollos empleando Spring como framework base, nos encontramos con anotaciones sobre las clases que desarrollamos; dentro de las principales anotaciones utilizando MVC no encontramos con 5, las cuales de alguna manera Spring encuentra, y nos permite interactuar con sus métodos sin la necesidad de declarar una nueva instancia mediante la palabra reservada new. Esto como ya supondrán se debe a la inyección de dependencias que emplea Spring.
Antes de la versión 2.5 de Spring, cada uno de los beans que necesitábamos emplear dentro de nuestro proyecto debían de ser declarados de manera manual dentro de un archivo XML, generando un gran problema para el mantenimiento de un desarrollo o de un sistema ya funcional ya que difícilmente se trabaja con solo un par de beans dentro del mismo.
De ahí nace la inyección de depencias en Spring 2.5: empleando la anotación @Component, Spring automáticamente escanea y registra la clase como un Spring bean, evitando que nosotros tengamos que declararlo de manera manual.

Ahora bien, entonces ¿para que sirven las otras tres anotaciones? Como en todo lenguajede programación todo tiene su razón de ser, cada una de las otras cuatro anotaciones para Spring funcionan como @Component ya que son sub @Anotaciones de la misma (ver imagenes siguiente), permitiendo que sean manipuladas por el Spring AplicationContex, además de agregarle cierta funcionalidad específica a la clase dependiendo de la anotación que se emplee en la clase.

Jerarquía de anotaciones



Por ejemplo, en @Controller vamos a encontrar @RequestMapping, el cual nos va a permitir recibir de una manera y un comportamiento esperado peticiones HTTP, algo queno ocurriría

si empleamos la @Component en lugar del @Controller.

Otro ejemplo, es la anotación @Repository, ya que se encarga de manejar excepciones específicas de una base de datos y devolverlas como una excepción estándar de Spring, permitiendo atender a las excepciones de una manera menos restrictiva.

Cuando usar cada una de las @Anotaciones

- **@Component:** anotación generica que emplea cualquier componente o beandentro de Spring.
- **@Repository:** anotación que debe ser empleada en la capa de persistencia.
- **@Service:** anotación que debe de ser empleada para la capa de servicios (facade) ylogica de negocio.
- **@Controller:** anotación que debe de ser empleada en la capa de presentación(Spring MVC).
- **@RestController:** anotación que debe ser empleada en la capa de presentación,para peticiones de tipo REST.
- **@Autowired:** se emplea para generar la inyección de dependencia a de un tipo deObjeto que pertenece a una clase con la @Component.

Qué es un Bean en Spring y cómo definirlo

A diferencia de los bean convencionales que representan una clase, la particularidad de los beans de Spring es que son objetos creados y manejados por el contenedor Spring.

Configurando nuestros Beans

Ahora que ya tenemos configurado el acceso al servlet central, debemos configurar los beans para atender los diferentes request. Spring, al inicializar el servlet DispatcherServlet, buscará un archivo con el nombre [servlet-name]-servlet.xml para localizar la información de configuración de nuestros beans. El nombre del archivo de configuracion de los beans puede ser cambiado mediante el parametro de inicialización del servlet, contextConfigLocation, indicando la ubicación del mismo.

La interfaz utilizada por el DispatcherServlet para la instanciación de los beans es WebApplicationContext. Dicha interfaz es una extension de ApplicationContext, pero posee

caracteristicas especiales necesarias para aplicaciones web.

Como obtener los beans

Cada vez que llega un request, para que podamos utilizar el webapplicationcontext, el DispatcherServlet agrega al request un atributo bajo el nombre

DispatcherServlet.WEB_APPLICATION_CONTEXT_ATTRIBUTE, conteniendo una referencia al webapplicationcontext.

```
WebApplicationContext appContext =
    (WebApplicationContext)  request
    .getAttribute(DispatcherServlet
    .WEB_APPLICATION_CONTEXT_ATTRIBUTE);

UnBean unBean = (UnBean) appContext.getBean("unBean");
```

Tipos especiales de beans en WebApplicationContext

Spring DispatcherServlet utiliza beans especiales para procesar las solicitudes y representar las vistas adecuadas. Estos beans son parte de Spring MVC, y es posible elegirqué beans utilizar simplemente configurando uno o más de ellos en elWebApplicationContext; sin embargo, no es necesario hacer que Spring MVC mantenga una lista de los beans a utilizar si no se configura ninguna. Antes de profundizar en este tema se mostrará una lista de los tipos de beans especiales en los que se basa el DispatcherServlet.

Tipo de Beans	Explicación
HandlerMapping	Mapa de peticiones entrantes a los controladores y una lista de pre-y-post-procesadores en base a unos criterios cuyos detalles varían según la implementación del HandlerMapping.
HandlerAdapter	Ayuda al DispatcherServlet a invocar un controlador asignado a la petición.
HandlerExceptionResolver	Mapas de excepciones para vistas, lo que permite un fácil manejo de excepciones más complejas.
ViewResolver	Resuelve los nombres de vistas basándose en cadenas lógicas dependiendo del tipo de las vistas.
LocaleResolver & LocaleContextResolver	Obtiene la configuración regional que un cliente está utilizando y su zona horaria, con el fin de ser capaz de ofrecer vistas internacionalizadas.

ThemeResolver	Resuelve temas que la aplicación web puede utilizar, por ejemplo, para ofrecer diseños personalizados.
MultipartResolver	Analiza las solicitudes que se dividen en varias partes, por ejemplo, para apoyar la carga de archivos de procesamiento de formularios HTML.
FlashMapManager	Almacena y recupera la "entrada" y "salida" FlashMap que se utilizan para transferir los atributos de una solicitud a otra, por lo general, a través de una redirección.

Configuración por defecto del DispatcherServlet

Tal como se mencionó anteriormente, para cada bean especial del DispatcherServlet se mantiene una lista de las implementaciones a usar por defecto. Esta información se guarda en el archivo DispatcherServlet.properties dentro del paquete org.springframework.web.servlet.

Todos los beans especiales tienen valores por defecto, aunque, tarde o temprano, habráque personalizar alguna de las propiedades que estos beans proporcionan. Por ejemplo es muy común para configurar un InternalResourceViewResolver y cambiar su propiedad"prefix" por el directorio de los archivos de la vista.

Una vez que se configura un bean especial, como un InternalResourceViewResolver en el WebApplicationContext, se reemplaza la lista de implementaciones predeterminadas quese han utilizado para ese tipo de bean especial. Por ejemplo, si se configura un InternalResourceViewResolver, se ignorará la lista de implementaciones predeterminadas de ViewResolver.

Secuencia de procesamiento del DispatcherServlet

Después de configurar un DispatcherServlet, cuando llega una petición para el DispatcherServlet, este empieza a procesar la solicitud de la siguiente manera:

- Se busca el WebApplicationContext y se consolidará la solicitud como un atributo que el controlador y otros elementos en el proceso pueden usar. Esta se asocia por defecto en la clave DispatcherServlet.WEB_APPLICATION_CONTEXT_ATTRIBUTE.
- La resolución de la configuración regional se une a la petición para permitir que los elementos obtengan la localización del proceso que se utilizará al procesar la solicitud (la representación de la vista, la preparación de los datos,etc.).
- Si no se utiliza resolución de la configuración regional este paso se omite.
- La resolución del temas se une con la petición para permitir que los elementos,como las vistas, determinen qué tema usar. Si no se utiliza resolución de temaseste paso se omite.
- Si especifica una resolución de archivo de varias partes, la solicitud lo inspecciona; si se encuentran varias partes, la solicitud se envuelve en un MultipartHttpServletRequest para su posterior procesamiento por otros elementos del proceso.
- Se busca un controlador adecuado. Si se encuentra un controlador, la cadena de ejecución asociada con el controlador (pre-procesadores, post- procesadores y controladores) se ejecuta con el fin de preparar un modelo o vista.
- Si se devuelve un modelo, la vista se representa. Si no se devuelve ningún modelo, (puede ser debido a que un pre-procesador o post-procesador intercepte la petición, tal vez por razones de seguridad), la vista no se representa, porque la petición ya se ha cumplido.

La resolución de excepciones de los controladores es declarada en el WebApplicationContext, y recoge excepciones que se producen durante el procesamiento de la solicitud. El uso de estos solucionadores de excepciones permite definir comportamientos personalizados para abordar las excepciones.

Spring DispatcherServlet también permite la devolución de la última fecha demodificación, como se especifica en el API Servlet. El proceso para determinar la última fecha de modificación de una solicitud específica es sencillo: el DispatcherServlet busca ycomprueba si el controlador implementa la interfaz LastModified. Si es así, el valor del método getlastmodified (request) de la interfaz LastModified se devuelve al cliente.

También se puede personalizar la instancia del DispatcherServlet añadiendo los parámetros (init-param) de inicialización a la declaración de servlets en el archivo web.xml.

La tabla siguiente muestra la lista de parámetros admitidos.

Parámetro	Explicación
contextClass	Clase que implementa WebApplicationContext, lo que crea una instancia del contexto utilizado por este servlet. Por defecto, se utiliza el XmlWebApplicationContext.
contextConfigLocation	Cadena que se pasa a la instancia del contexto (especificado por contextClass) para indicar dónde se pueden encontrar los contextos. La cadena consiste potencialmente de varias cadenas (utilizando una coma como delimitador) para soportar múltiples contextos. En caso de utilizar distintos contextos con los beans que se definen en dos ocasiones, la última ubicación tiene prioridad.
Espacio de nombres	Espacio de nombres de la WebApplicationContext. El valor predeterminado es (servlet-name)-servlet.

Creando un proyecto web Spring MVC

Lo primero que hay que entender es que Spring es un gran todo que engloba muchas partes del desarrollo, no sólo web, sino Java en general; vamos a utilizar dos de esas partes, que son:

Spring Container: Es la base de Spring. Podemos verlo como una bolsa de Java Beans que residen, en este caso al ser una aplicación web, en nuestro contenedor de aplicaciones (normalmente nuestro servidor web, por ejemplo, Tomcat). Lo más importante de todo es que no nos importa cómo funciona ni cómo gestiona nuestros beans, solamente que los pondrá a nuestra disposición desde (casi) cualquier parte de nuestro código.

Spring MVC: La implementación Spring del patrón o pattern MVC, hoy por hoy el más extendido, por muchas buenas razones, para el desarrollo web.

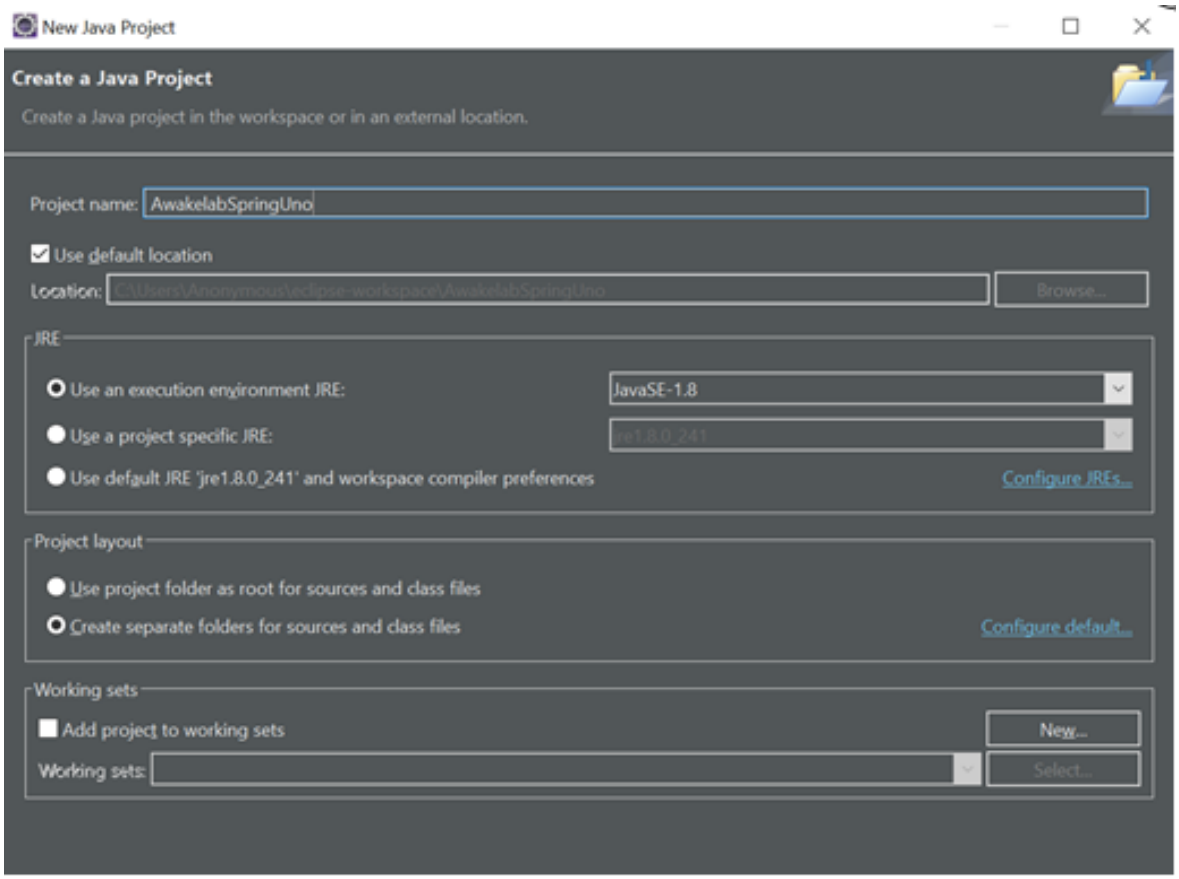
Spring provee diversas opciones para configurar tu aplicación. La forma más tradicional es usando archivos XML, pero en la actualidad se dispone de una manera alternativa de configurar nuestras aplicaciones Spring. Las nuevas versiones de Spring introducen un amplio soporte para configurar una aplicación web mediante anotaciones y para el

desarrollo acelerado de aplicaciones mediante Spring Boot. Este documento usa el estilomás moderno de configuración mediante anotaciones.

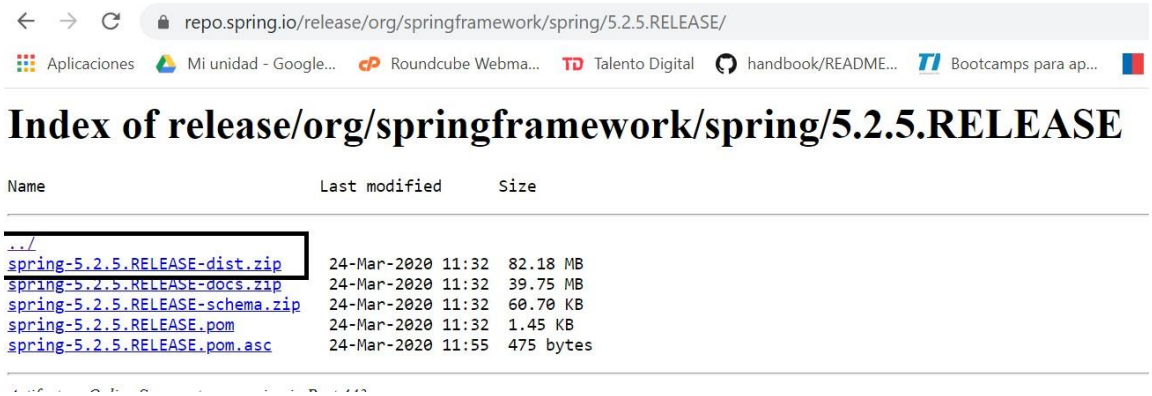
El proyecto Spring Tool Suite (https://spring.io/tools) proporciona un excelente entorno para el desarrollo de aplicaciones que utilicen Spring Framework. Spring Tool Suite se integra sobre la plataforma Eclipse (http://www.eclipse.org/) y también da soporte al servidor web (en nuestro caso Tomcat); sin embargo, podemos preparar el entorno desdeel IDE Ecplise con las librerias necesarias.

Creación de proyecto Spring MVC – Forma convencional

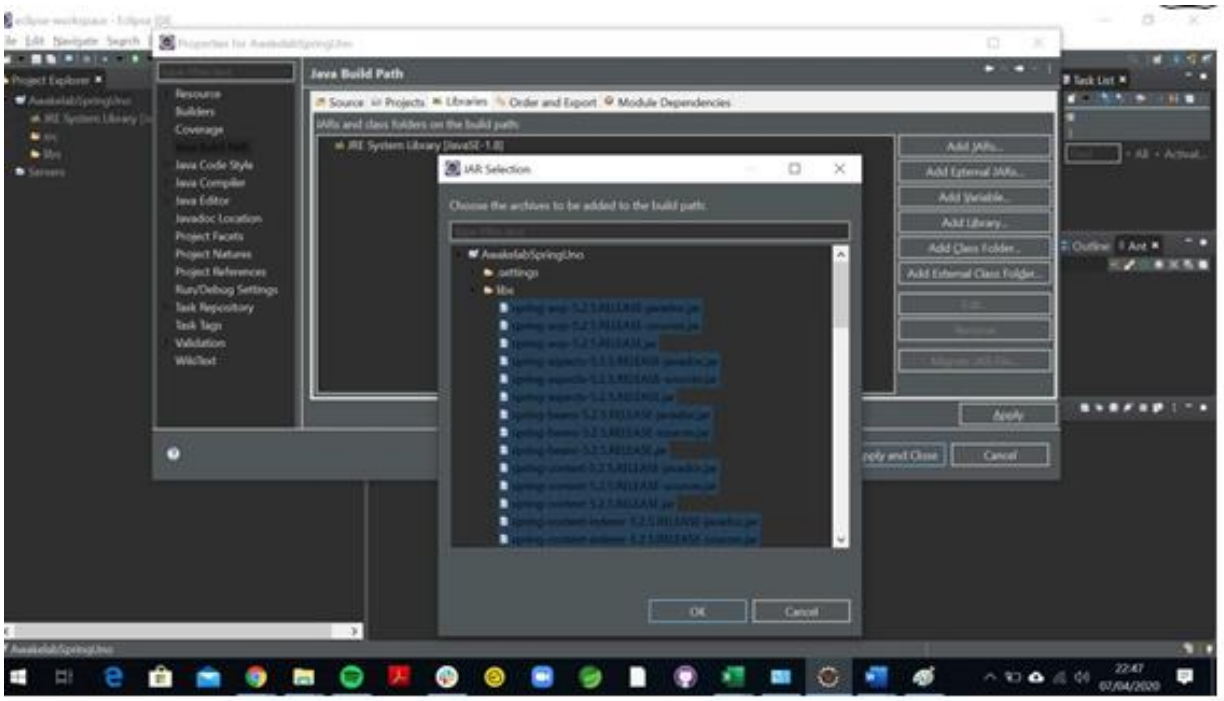
1. Crearemos un proyecto Java, al cual se deberán importar las librerías .jarnecesarias para el funcionamiento del entorno.



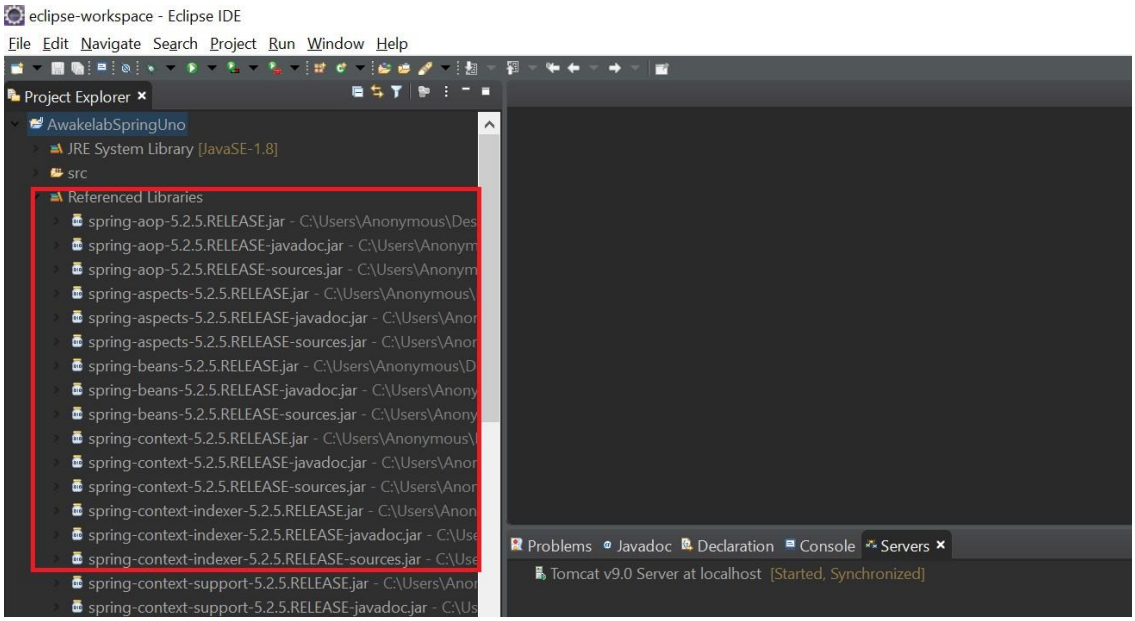
2. Descarga desde https://repo.spring.io/release/org/springframework/spring/ la ultima version de los spring jar files necesarios para el proyecto Spring; después se descomprime el archivo zip, en donde encontrarás la carpeta libs con las respectivas jars.



3. Como ultimo paso, debemos indicar en el poyecto las librerías a utilizar; paraello hacemos click derecho en el proyecto > Build Path > Configure Build > “Add External JARs”. Desde el directorio descomprimido, importa las libreríasSpring necesarias para el proyecto.

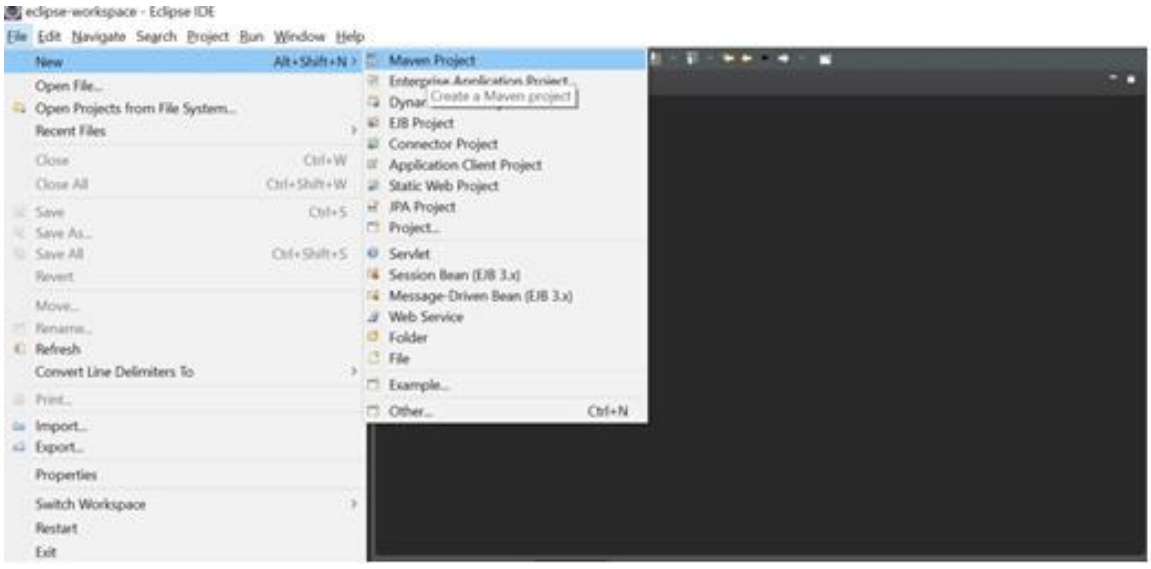


4. Con este último paso, el entorno se encuentra preparado para la creación deun proyecto básico en Spring.

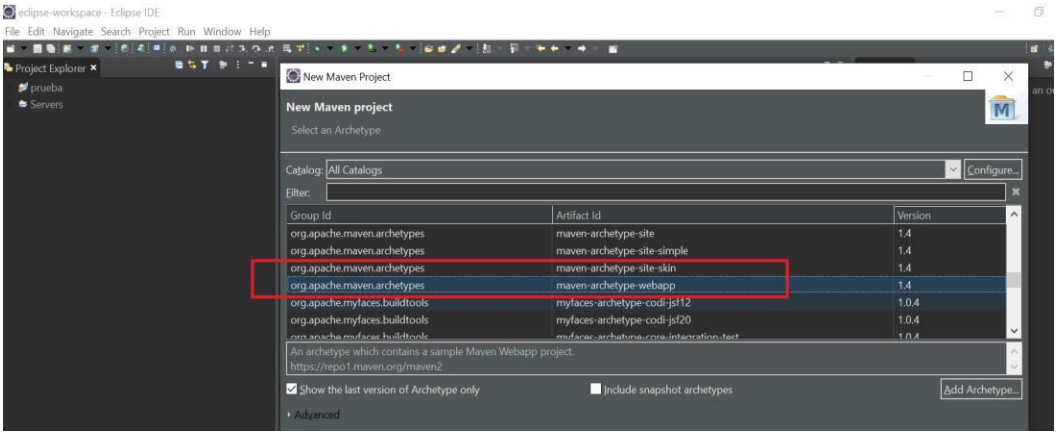


2.4.2.- Creación de proyecto Spring MVC – Gestión de dependencias con Maven

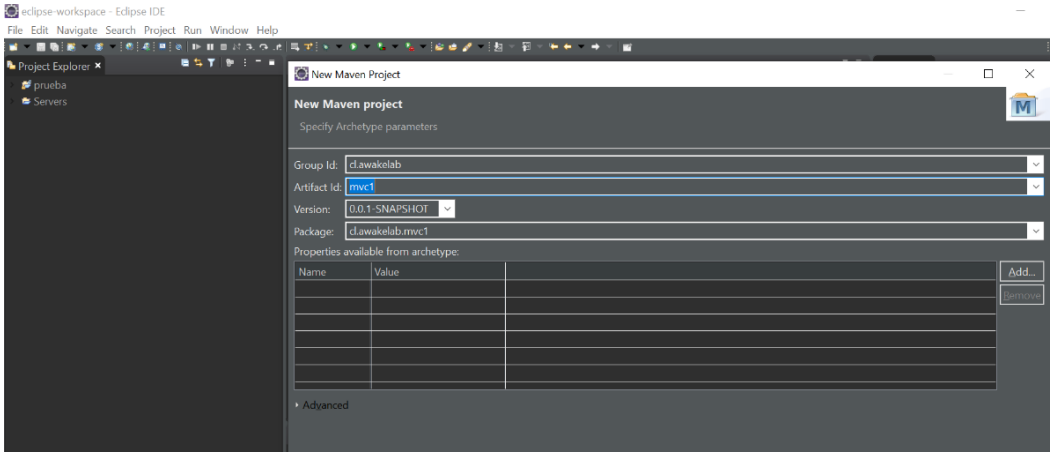
1. Para crear un nuevo proyecto, iremos al menú principal, y seleccionaremos laopción File > New > Maven Project.



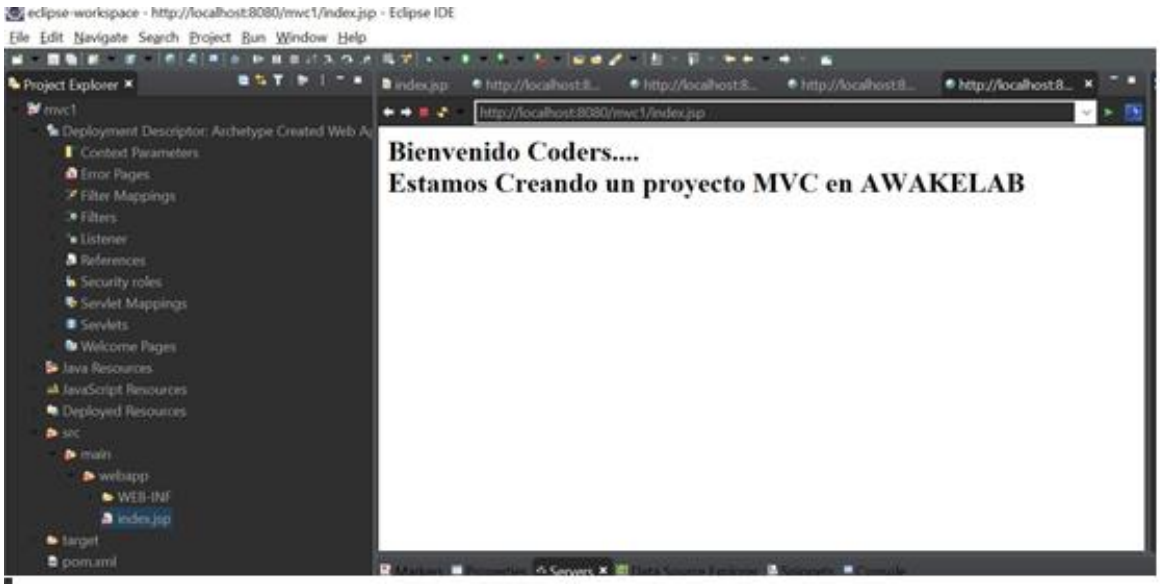
2. En la vista de catálogos, seleccionamos el ítem maven-archetype-webapp.



3. Indicamos el nombre del Goup Id y Artefact Id, damos click en Finalizar yesperamos a que se cree la estructura del proyecto.

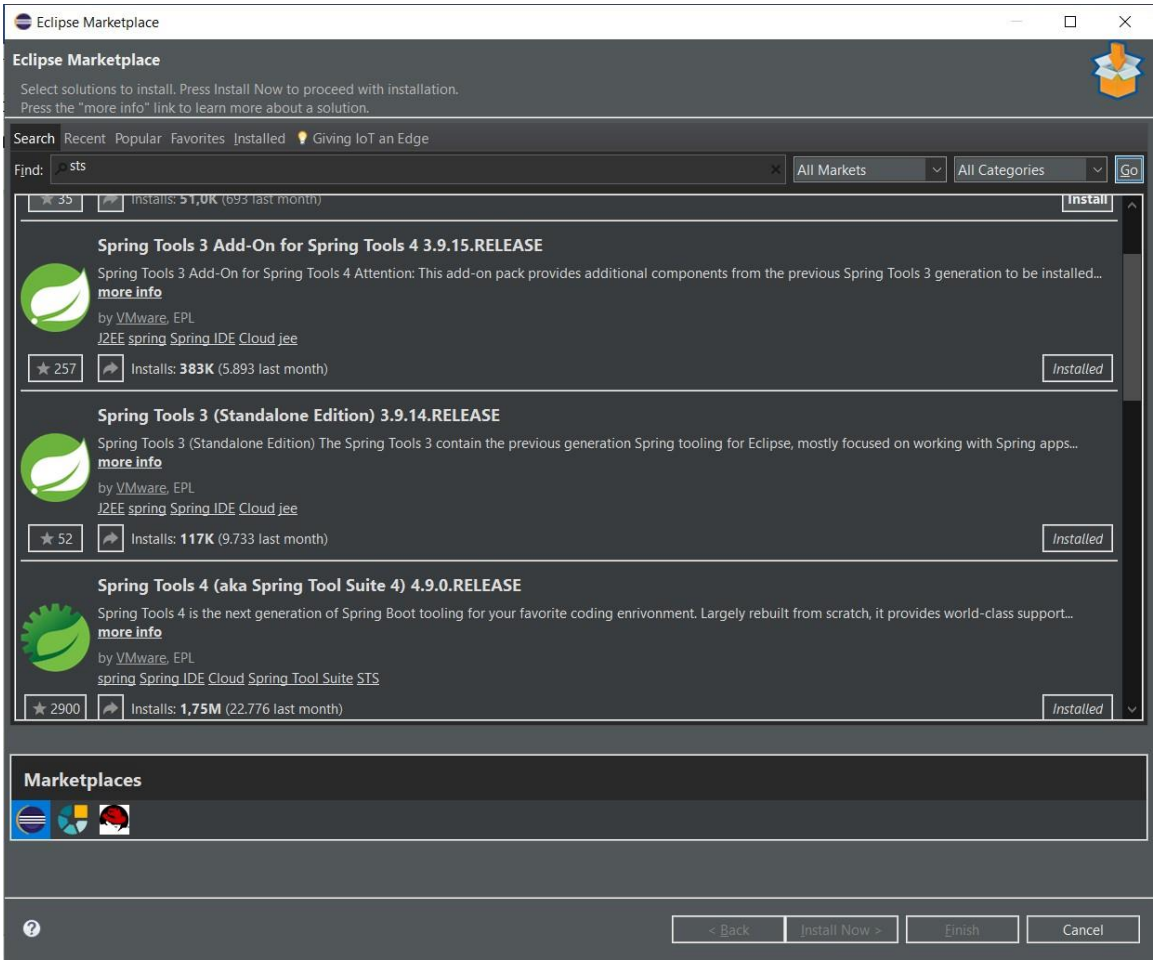


4. Finalmente podemos revisar que la estructura se ha creado correctamente yejecutar el index.jsp para verificar el funcionamiento de nuestro entorno.

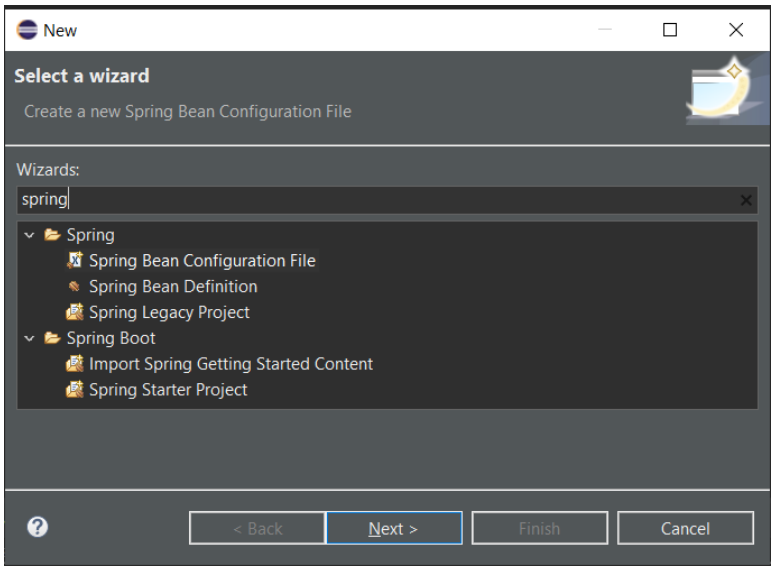


Creación de proyecto Spring MVC – Spring Tool Suite

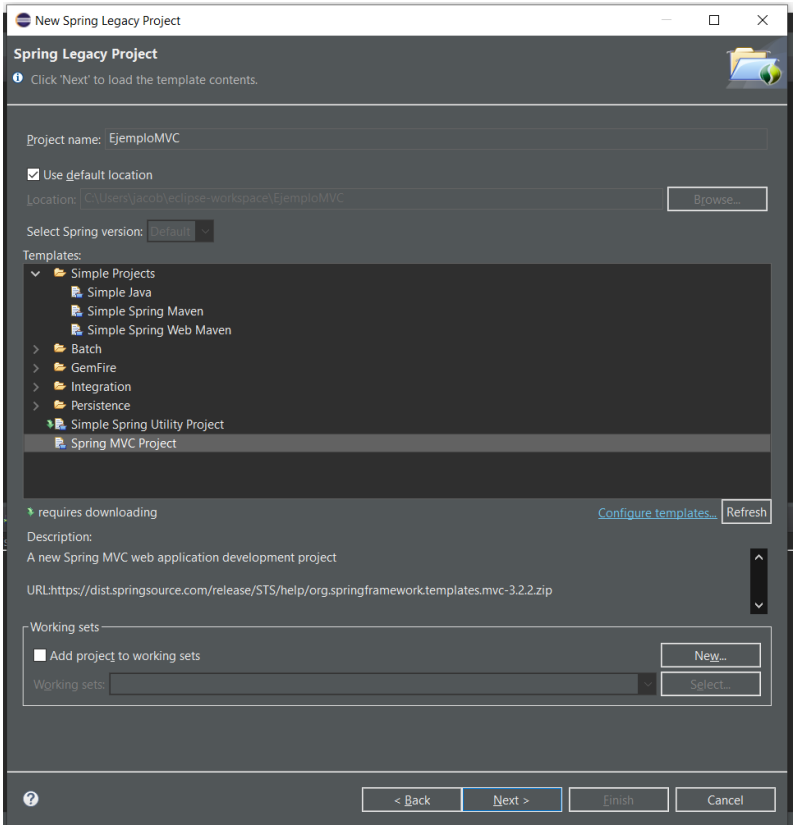
1. Primero que todo, es necesario que tengas instalada la extensión Spring Tool Suite 3 en tu Eclipse. Si no lo ha hecho, en el menú principal selecciona la opción Help → Eclipse Marketplace , y en el cuadro de búsqueda colocala palabra “STS”. Presionando el botón “Go” se desplegará un listado comoel siguiente (si el botón ubicado al costado derecho dice “Install”, hace clic enél):



2. Para crear un proyecto Spring MVC a través de esta herramienta, anda al menú principal en la opción “File → New → Other”. En la casilla de búsqueda escribe la palabra Spring, selecciona la opción “Spring Legacy Project” y presionael botón “Next”.

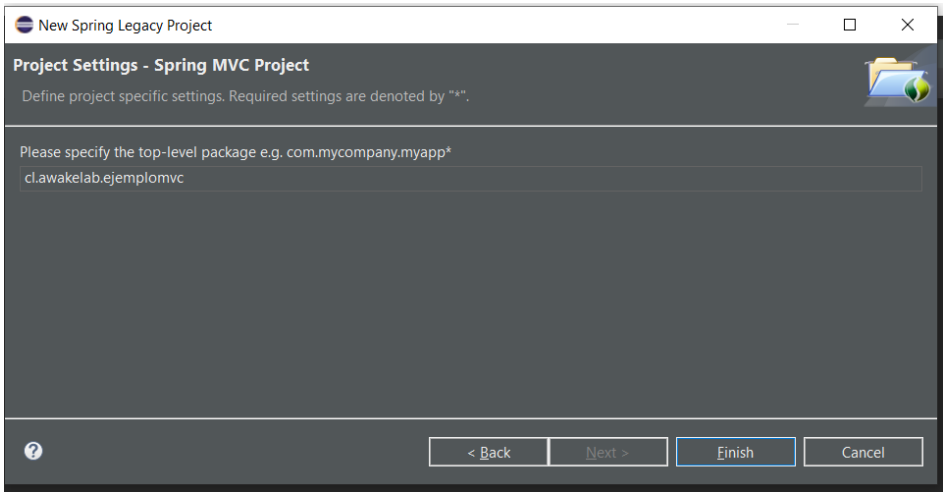


3. Coloca un nombre al proyecto, selecciona la opción “Spring MVC Project” ypresiona “Next”.

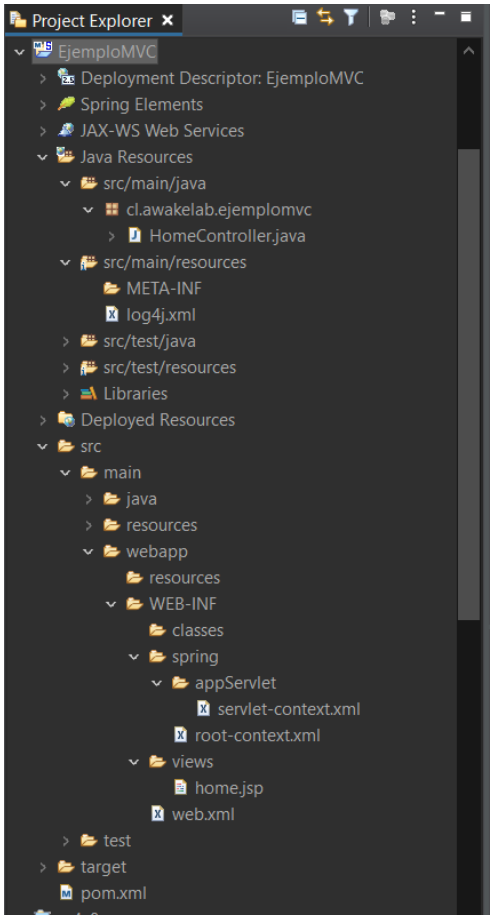


4. En la ventana que aparecerá, indica el formato de nombre de paquetes de la

aplicación. Se debe respetar el formato que ahí se indica; una vez realizado presiona el botón “Finish”.



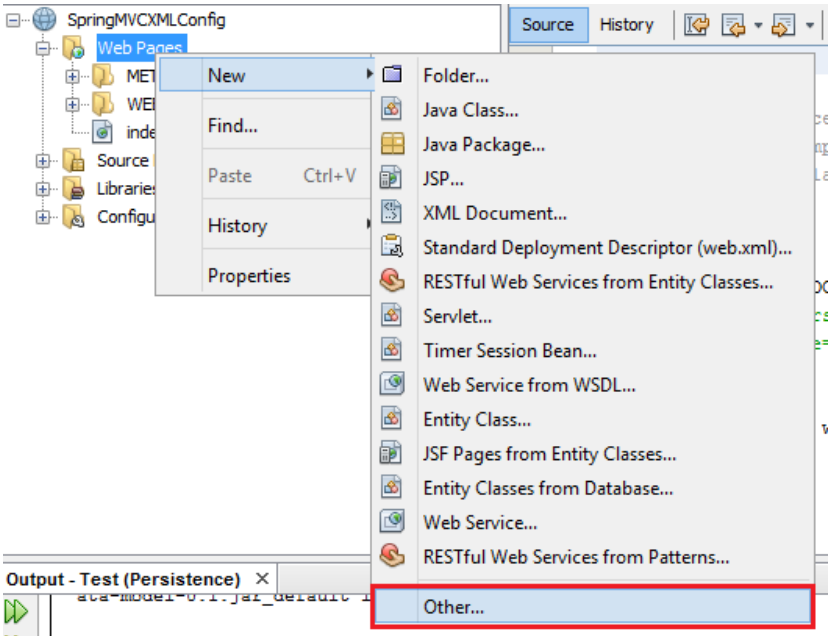
5. Con ello se creará un proyecto Spring MVC, con manejo de dependencias a través de Maven. Bajo esta lógica, gran parte de las configuraciones que se indican en el punto siguiente no aplican, ya que la plataforma lo definirá de forma automática.



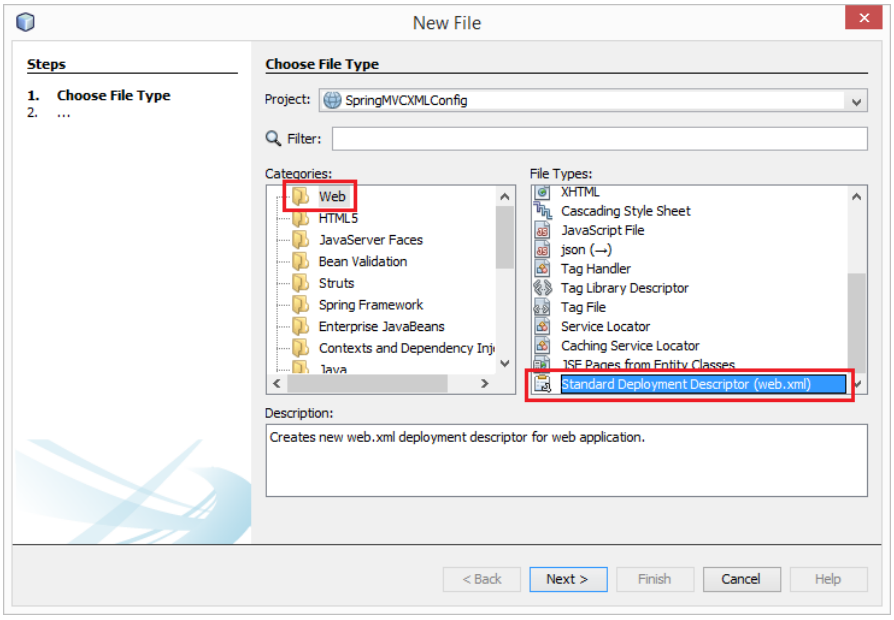
Configuraciones iniciales de Spring MVC

Antes de continuar, es necesario destacar que gran parte de esta sección solo aplica en caso que se haya generado un proyecto Spring MVC sin usar STS. Sin embargo, igualmenteayuda a entender qué rol cumple cada archivo del proyecto.

En este caso para configurar el "DispatcherFilter" de Spring MVC necesitamos un deployment descriptor o archivo "web.xml" de nuestro proyecto web. Para agregar este archivo hacemos clic derecho en el proyecto y seleccionamos "New -> Other...".



En la categoría elegimos "Web" y en "File Types" seleccionamos "Standard Deployment Descriptor (web.xml)"



Presionamos el botón "Finish", con lo veremos aparecer el contenido de nuestro archivo.

En este archivo configuraremos el DispatcherServlet que, como ya dijimos, es la pieza central de Spring MVC. Este Servlet procesará todas las peticiones que vayan hacia el framework, y decidirá qué componente debe atender cada una de las solicitudes.

A este Servlet podemos darle cualquier nombre que queramos, y la clase que lo implementa es "org.springframework.web.servlet.DispatcherServlet". Este Servlet debe ser el primero en iniciar cuando se cargue la aplicación, por lo que deberá ser el número 1 en el orden de inicio. La configuración del "DispatcherServlet" queda de la siguiente forma:

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

Lo siguiente es indicar qué patrón de peticiones serán las que pasen a través del DispatcherServlet; en nuestro caso queremos que todas las peticiones pasen a través de él, por lo que la configuración queda de esta forma:

```
<servlet-mapping>
    <servlet-name>spring-web</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
```

Como podemos ver, para configurar el patrón de URLs que se enviarán a este Servlet debemos indicar, primero, el nombre que habíamos mencionado anteriormente, y como patrón de URLs la cadena "/".

Nuestro archivo "web.xml" completo se verá de la siguiente forma:

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
         version="3.1">

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

```
    <servlet-mapping>
        <servlet-name>spring-web</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <session-config>
        <session-timeout>30</session-timeout>
    </session-config>
</web-app>
```

Lo siguiente es configurar el "DispatcherServlet" para indicar algunos datos particulares de cómo queremos que se comporte. El "DispatcherServlet" trae por default un conjuntode configuraciones que sirven para la mayoría de los casos. En nuestro caso la única configuración que haremos es cómo mapear los nombres lógicos de las respuestas a las páginas que conforman las vistas.

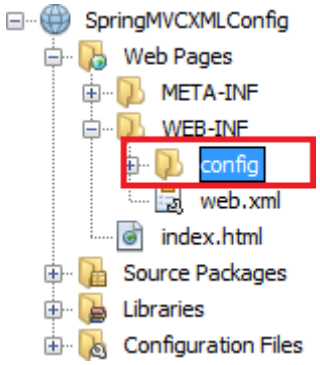
Esta configuración debe estar en un archivo XML que debe colocarse a la misma altura que el archivo "web.xml" – o sea en el directorio "WEB-INF" de la aplicación. El nombre de este archivo debe seguir una convención, y debe ser el mismo nombre que le hayamosdado al "DispatcherServlet" en la configuración del Deployment Descriptor (en nuestro caso "spring-web"), agregando "-servlet.xml", por lo que en mi caso el nombre del archivodebe ser "spring-web-servlet.xml".

¿Qué ocurre si queremos colocar el archivo de configuración de Spring MVC en otra ubicación o queremos darle otro nombre? Para esto existe una manera de sobre-escribirla ruta del archivo, a través de un parámetro que pasamos a la configuración del "DispatcherServlet". Es recomendable tener estos archivos de configuración en un directorio "config" dentro del "WEB-INF". Para lograr esto agregamos un parámetro llamado "contextConfigLocation", cuyo valor será la ubicación del archivo de configuración de Spring MVC. En este caso el nombre del archivo será "springMVCconfig.xml", y estará en el directorio "WEB-INF/config".

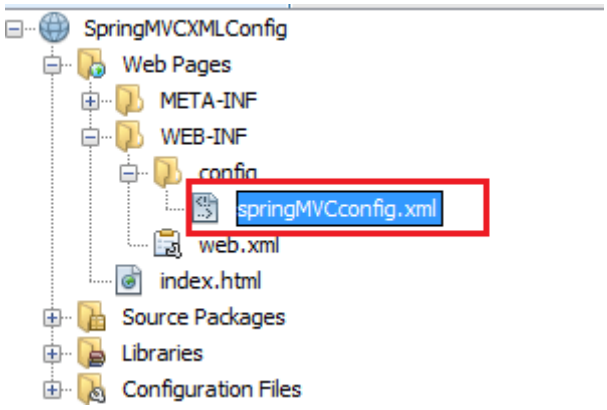
Modificamos la configuración del "DispatcherServlet" para incluir este parámetro, de la siguiente forma:

```
<servlet>
    <servlet-name>spring-web</servlet-name>
    <servlet-class>
        org.springframework.web.servlet.DispatcherServlet
    </servlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            /WEB-INF/config/springMVCconfig.xml
        </param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
```

Ahora, creamos un nuevo directorio llamado "config" dentro de "WEB-INF":



Y dentro de este creamos un archivo XML llamado "springMVCconfig.xml":



Inicialmente el contenido del archivo será el siguiente:

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-
4.2.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-
4.2.xsd">
```

Usamos el elemento "component-scan" del namespace "context" para indicar dónde se encuentran los componentes de Spring; recuerda que este elemento habilita los elementos que tengan las anotaciones @Component, @Repository, @Service, y @Controller (en nuestro caso sólo usaremos esta última). Usamos el atributo "base- package" para indicar en qué paquete están nuestros controladores, en nuestro caso "com.javatutoriales.springmvc.configuracion.controllers". Este elemento queda de la siguiente forma:

```
<context:component-scan
base-package="com.javatutoriales.springmvc.configuracion.controllers"
/>
```

Lo siguiente es habilitar los componentes de Spring MVC con sus configuraciones por defecto; esto lo hacemos con el elemento "annotation-driven" del namespace "mvc", dela siguiente forma:

```
<mvc:annotation-driven />
```

Esta etiqueta adicionalmente registrará los "HandlerMapping" y "HandlerAdapter" requeridos para despachar los Controllers. Adicionalmente, aplica algunas configuraciones por default basado en lo que se encuentra en nuestro classpath. Estas configuraciones por default son:

- Agrega soporte para formatear campos numéricos anotados con @NumberFormat.
- Agrea soporte para formateo de campos tipo Date, Calendar, y Time anotados con @DateTimeFormat.
- Agrega soporte para validar campos de entrada en @Controller con @Valid, si un proveedor JSR-303 se encuentra en el classpath.
- Agrega soporte para leer y escribir XML, si JAXB se encuentra en el classpath.
- Agrega soporte para leer y escribir JSON, si Jackson se encuentra en el classpath.

Finalmente debemos indicar qué implementación de "ViewResolver" usaremos. En este caso será un "InternalResourceViewResolver", el cual permite indicar el nombre lógico dela página que representa la vista, siempre y cuando el nombre del archivo sea igual al nombre lógico de nuestra vista; esto quiere decir que si decidimos que nuestra vista seránJSP y que estarán en el directorio "WEB-INF/pages", y un Controller indica que se debe regresar una vista llamada "accesoUsuario", esto quiere decir que debe existir un archivo "/WEB-INF/pages/accesoUsuario.jsp".

Este "ViewResolver" se configura como un bean "normal" de Spring, de la siguiente forma:

```
<bean
class="org.springframework.web.servlet.view.InternalResourceViewResolv
er">
</bean>
```

Para configurar el "InternalResourceViewResolver" debemos indicar cuál será el prefijo de las páginas. Este por lo general se usa para indicar en qué directorio se encuentran laspáginas (a partir de la raíz donde se colocan las páginas web), que en nuestro caso será en el directorio "/WEB-INF/pages", por lo que hasta ahora la configuración se ve de la siguiente forma:

```
<bean
class="org.springframework.web.servlet.view.InternalResourceViewResolv
er">
    <property name="prefix">
        <value>/WEB-INF/pages/</value>
    </property>
</bean>
```

La segunda propiedad que debemos indicar es el sufijo que tendrán las páginas. En esta propiedad por lo general indicamos la extensión de las mismas. En nuestro caso las páginas serán JSPs, por lo que el sufijo será ".jsp". La configuración queda de esta forma:

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

Y el archivo "springMVCconfig.xml" completo queda de la siguiente forma:

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:context="http://www.springframework.org/schema/context"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mvc="http://www.springframework.org/schema/mvc" xsi:schemaLocation="
        http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-
        beans-
        4
        .                                   http://www.springframework.org/schema/mvc
        2                                   http://www.springframework.org/schema/mvc/spring-mvc-4.2.xsd
        .                                   http://www.springframework.org/schema/context
        x                                   http://www.springframework.org/schema/context/spring-context-
        s
        d

        4.2.xsd">

            <context:component-scan base- package="com.javatutoriales.springmvc.configuracion.controllers" />

        <mvc:annotation-driven   />

            <bean class="org.springframework.web.servlet.view.InternalResourceViewResolv er">
        <property name="prefix">
        <value>/WEB-INF/pages/</value>
        </property>
        <property name="suffix">
        <value>.jsp</value>
        </property>
        </bean>
        </beans>
```

Ya tenemos todos los elementos en su lugar, por lo que el siguiente paso es ejecutar la aplicación y entrar a la siguiente dirección: http://localhost:8080/holaspringmvc, con lo que debemos de ver la siguiente pantalla:

*Esto indica que todo está bien configurado y funciona correctamente

Anexo: Referencias

[1] Spring MVC: Configuración
Referencia: https://www.javatutoriales.com/2015/12/spring-mvc-parte-1-configuracion.html

[2] Introducción a MVC en Spring
Referencia: http://www.jtech.ua.es/j2ee/publico/spring-2012-13/sesion03-apuntes.html

[3] Crear un proyecto MVC con Spring Tool Suite (STS)
Referencia: https://www.youtube.com/watch?v=eMG9qi061D8