Spring Framework

¡Comencemos!

¿Qué aprenderás?

- Conocer el framework de Spring.
- Conocer Spring Tools Suite.
- Instalar Spring Tools Suite.
- Iniciar proyecto con Spring Tools Suite.

Introducción

Hoy en día, Spring al ser una herramienta modular permite crear aplicaciones de manera rápida y sencilla, además de tener muchos módulos en un mismo lugar, esto quiere decir que ya no se necesita conocer diferentes librerías o módulos, spring permite trabajar con ellos de manera casi nativa. Para muchos proyectos, se considera la columna vertebral de la aplicación, ya que al tener muchos módulos y librerías compatibles lo hace muy fácil de integrar.

Se explicará la importancia de un Framework en el desarrollo de aplicaciones rápidas y sencillas, además de conocer Spring Framework, que se considera una de las mejores herramientas de desarrollo de aplicaciones en el lenguaje de programación Java y su paso por convertirse en una herramienta modular de desarrollo más que en un Framework. También se guiará en cómo instalar Spring Tools Suite o como agregar en el IDE de Eclipse las funcionalidades de Spring.

¡Vamos con todo!

¿Qué es un Framework?

Un *framework*, es un entorno de trabajo que provee al desarrollador un conjunto de herramientas, módulos, librerías, metodologías y conceptos que permiten desarrollar un sistema informático de forma rápida. Un framework, permite reutilizar código reduciendo considerablemente los tiempos para la generación de aplicaciones.

Entonces, ¿Qué es Spring Framework?

Spring es un Framework de Java, que permite crear sistemas informáticos de alto rendimiento, livianos y reutilizables. Aportando herramientas necesarias para agilizar, estandarizar y resolver problemas que vayan apareciendo a medida que se crea la aplicación, evitando así la programación de tareas repetitivas.

En los inicios de Spring, era solo un framework para inyección de dependencias, hoy, ya se considera un sistema modular, es decir, posee diferentes módulos que agrupan distintas funcionalidades como, seguridad (Spring Security), servicios web (Spring Web Services), manejo de datos (Spring Data), android (Spring for Android), entre muchas otras.

En la siguiente imagen se observa todos los módulos que posee Spring, dentro de los cuales ya abarca gran parte del flujo de información y módulos para la creación de aplicaciones.

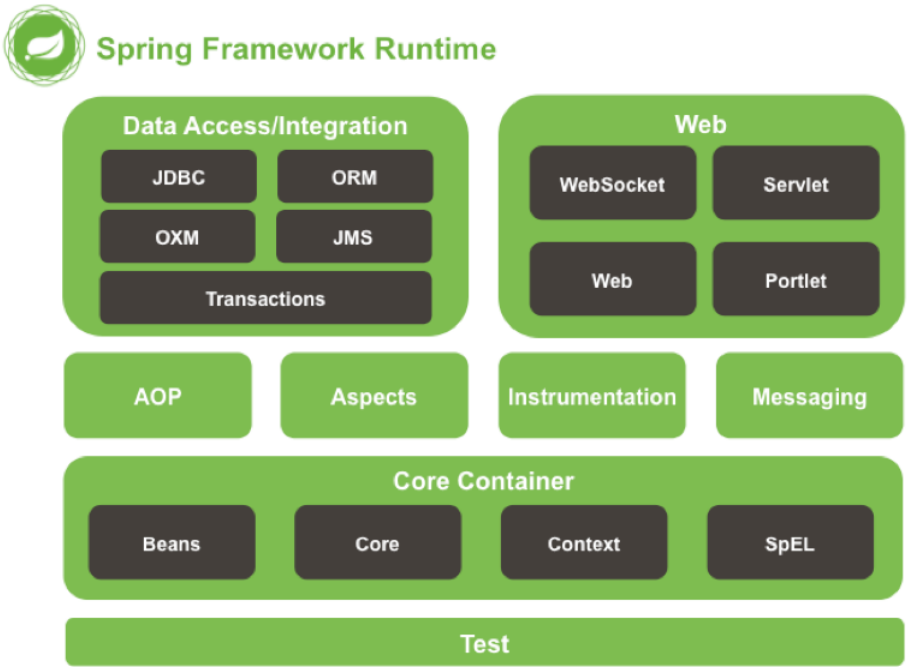

Imagen 1. Módulos de Spring.
Fuente: Desafío Latam.

Spring se considera el Framework padre de Java, ya que permite la integración con otras herramientas de forma natural, además de mantener el código ordenado en base a sus estándares de programación.

Spring utiliza el patrón de diseño de inyección de dependencias, lo que permite tener el código con implementación de servicios independientes.

Introducción a STS

Existen variados IDE en el mercado que nos permiten trabajar con Spring, como lo son NetBeans, Eclipse, IntelliJ Idea, entre otros.

Spring al ser hoy una herramienta, también generó su propio IDE, basándose en eclipse, para java, nombrandolo *Spring Tools Suite*, que es el que utilizaremos mayoritariamente en este curso.

Existen tres maneras para crear proyectos spring:

1. La primera es agregar spring tools mediante Eclipse Marketplace, en la opción de Ayuda.

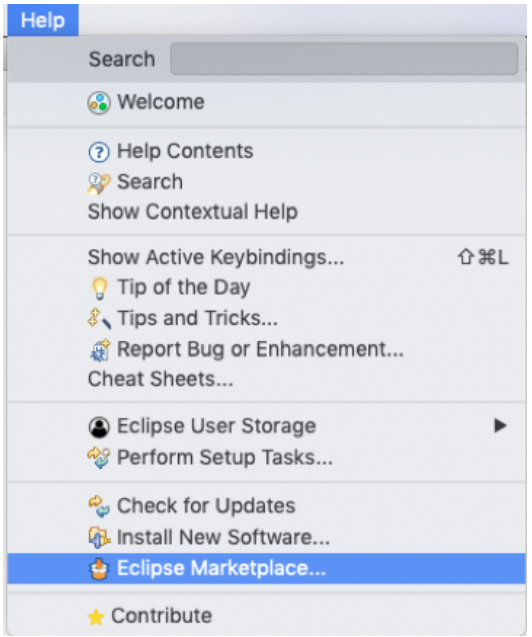

Imagen 2. Eclipse Marketplace.
Fuente: Desafío Latam.

Buscando Spring en la casilla para buscar, e instalando la versión de spring tools 4.

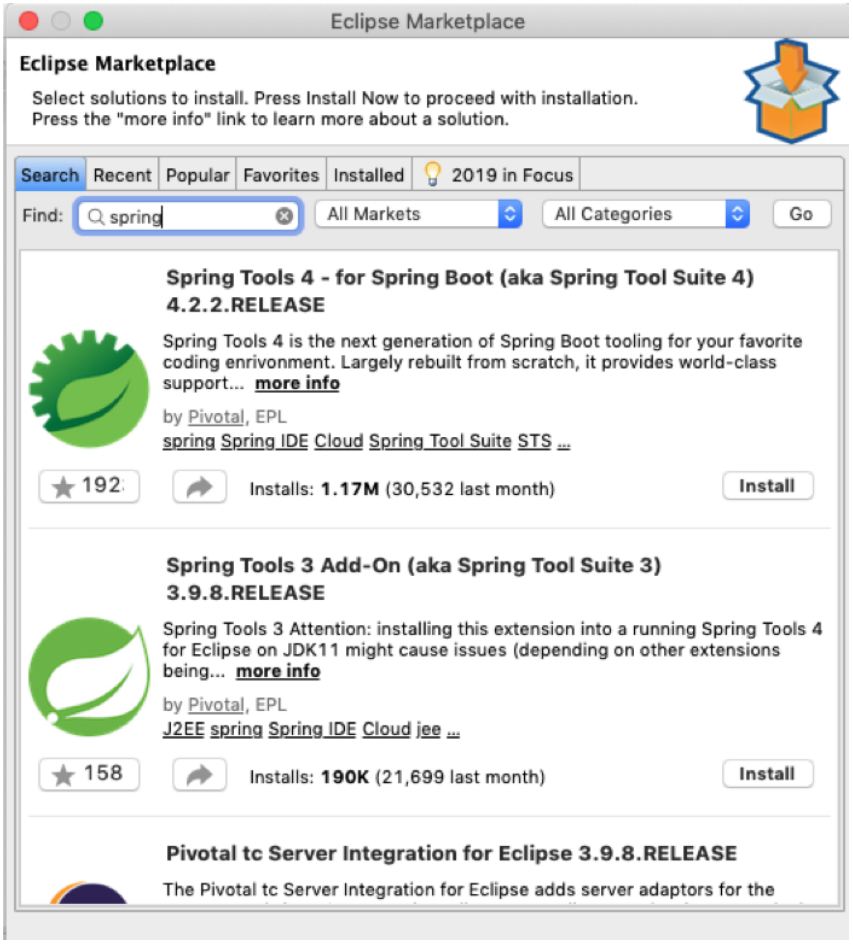

Imagen 3. Buscando Spring Tool 4.
Fuente: Desafío Latam.

Para verificar si se instaló correctamente, intentemos generar un proyecto y nos aparecerá la opción de spring.

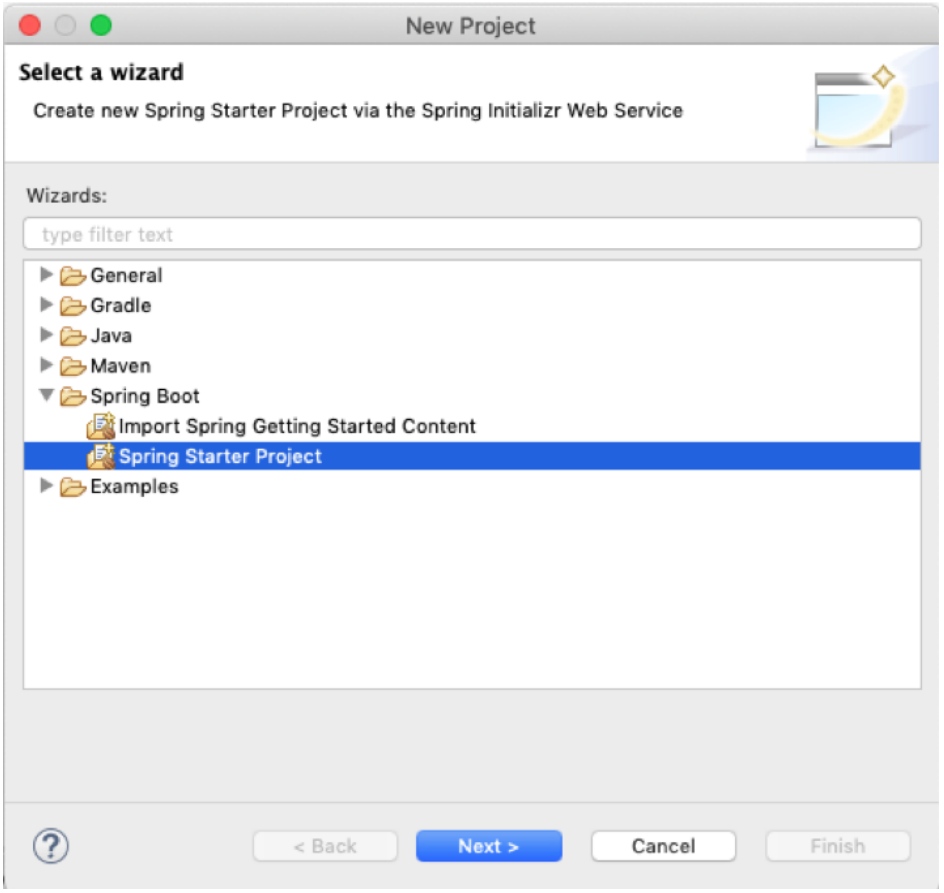

Imagen 4. Verificando la incorporación de Spring Tool 4.
Fuente: Desafío Latam.

2. La segunda opción, es descargarlo directo desde spring: Nos dirigimos al sitio.

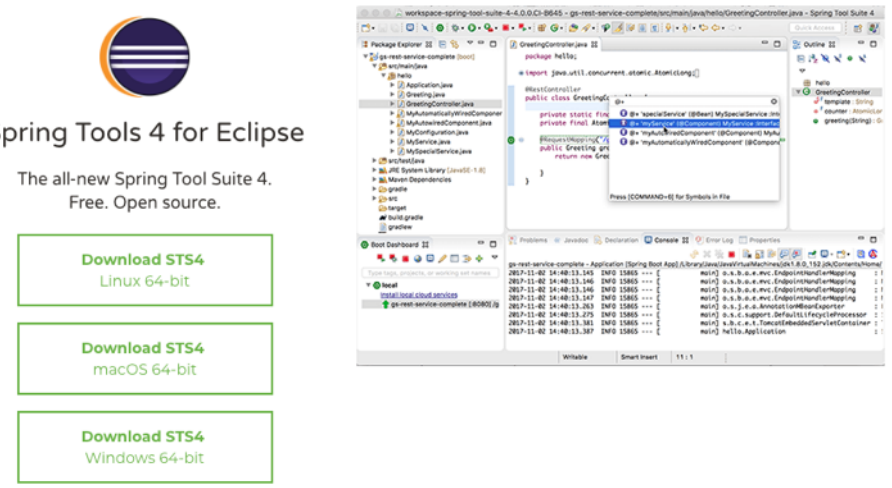

Imagen 5. Descargando STS.
Fuente: Desafío Latam.

Descargamos la opción que se acomode a nuestro equipo de trabajo y crearemos nuevo proyecto y vemos que nuestro el IDE ya muestra spring integrado, y se debería ver algo así:

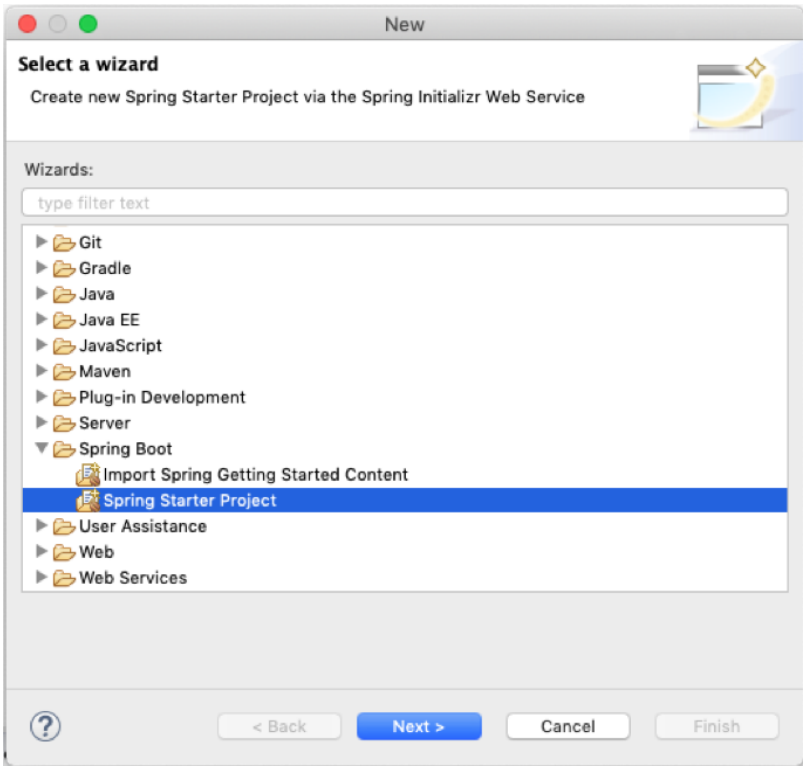

Imagen 6. Vista desde STS para crear un nuevo proyecto.
Fuente: Desafío Latam.

Seleccionamos el starter y tendremos que agregar nuestra información del proyecto, como es información de ejemplo, lo dejamos todo por defecto:

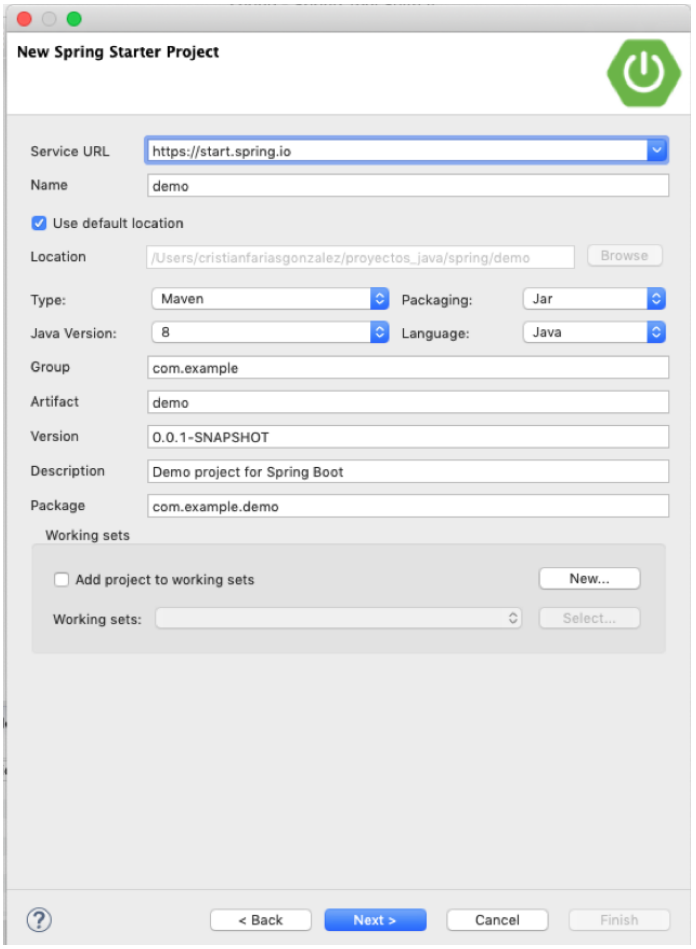

Imagen 7. Creando un nuevo proyecto.
Fuente: Desafío Latam.

Ahora, debemos seleccionar qué dependencias queremos cargar en nuestro proyecto, seleccionamos la opción de web, y finalizamos. También nos muestra dependencias como Spring HATEOAS o Jersey que nos sirven para implementar una API REST.

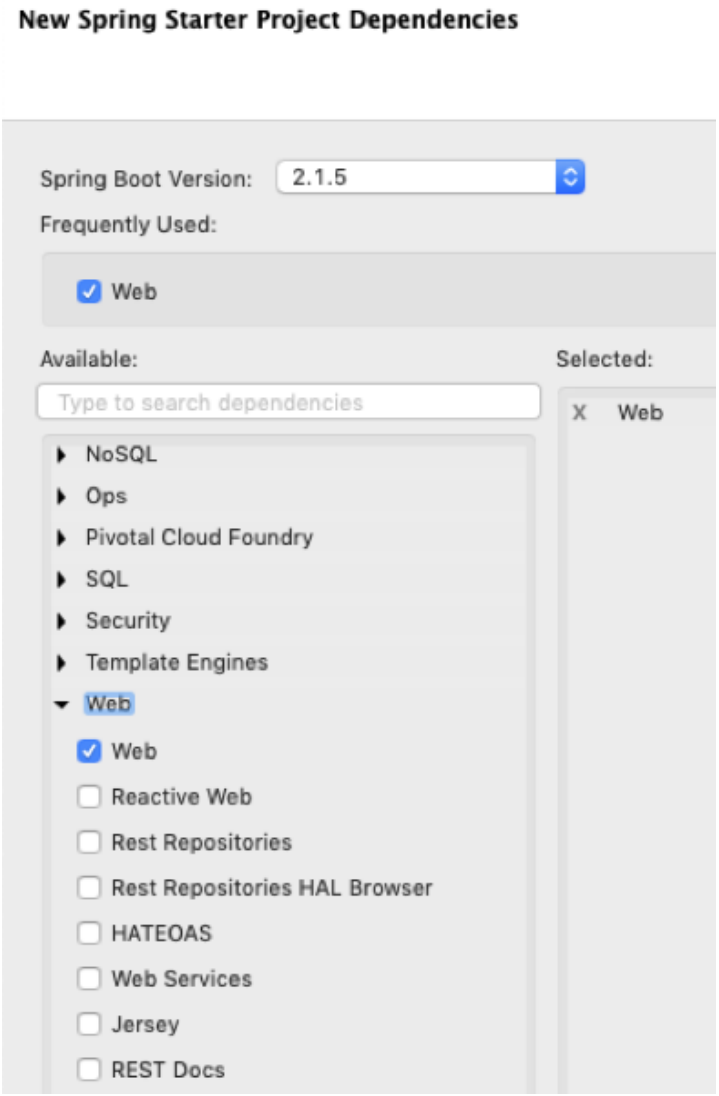

Imagen 8. Agregando dependencias.
Fuente: Desafío Latam

3. La tercera opción, que es descargar una herramienta spring lista, entonces, nos dirigimos a la url, y obtendremos una imagen como esta:

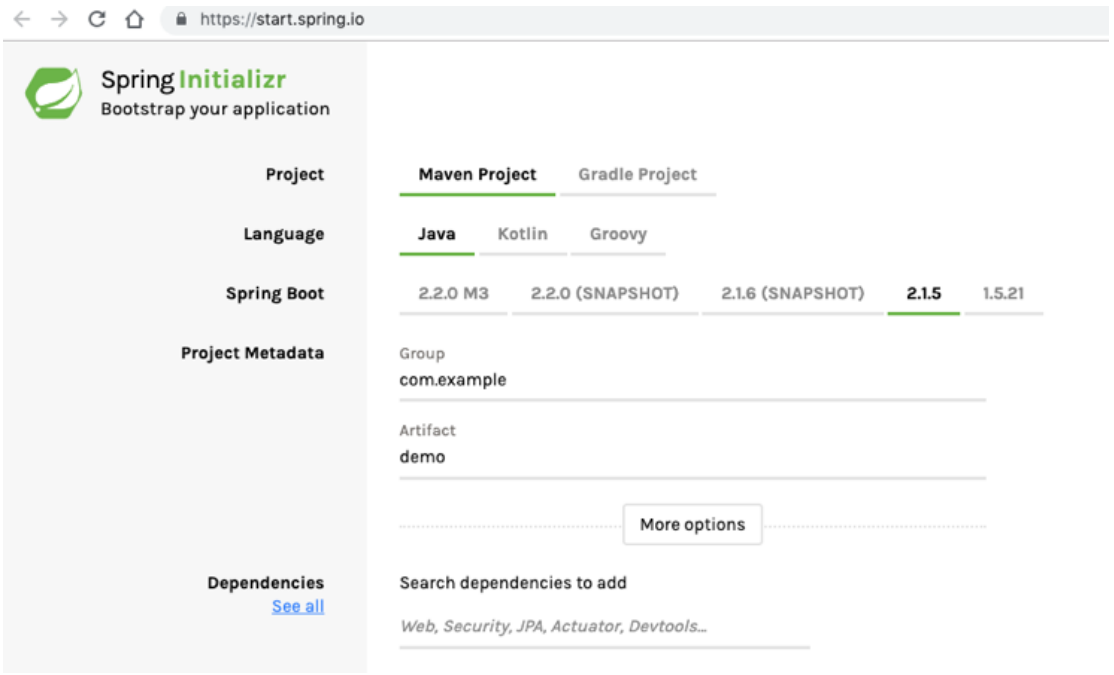

Imagen 9. Spring Initializr.
Fuente: Desafío Latam.

Simplemente, agregamos las dependencias necesarias y generamos el proyecto, se descargara un archivo con la estructura básica.

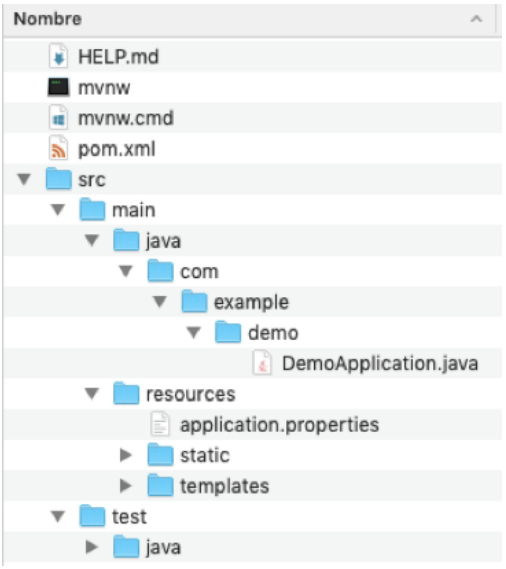

Imagen 10. Estructura básica del proyecto.
Fuente: Desafío Latam.

Utilizaremos el STS, al crear un proyecto spring, incorporando la dependencia de web, generará una estructura de directorio sencilla, muy parecida a la que vimos con maven.

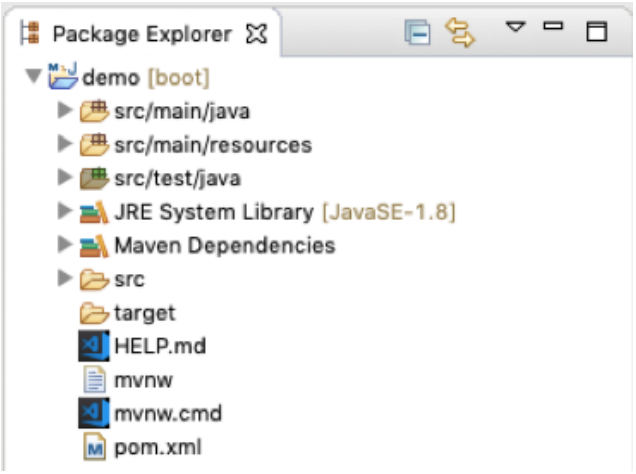

Imagen 11. Estructura creada con STS.
Fuente: Desafío Latam.

Si revisamos el archivo *POM.xml*, observamos que automáticamente la generación de proyectos spring, inserta las dependencias necesarias para el proyecto. Además otro tipo de información extra para manejar el artefacto de este proyecto.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>2.1.5.RELEASE</version>
      <relativePath/> <!-- lookup parent from repository -->
   </parent>
   <groupId>com.example</groupId>
   <artifactId>demo</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <name>demo</name>
   <description>Demo project for Spring Boot</description>
   <properties>
      <java.version>1.8</java.version>
   </properties>
```

```
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

Curiosidades del archivo POM.xml

Encontramos la etiqueta *<parent>*, esta indica que el proyecto que generamos, hereda todas las librerías y aplicaciones propuestas en el proyecto que hace referencia, en este caso, estamos haciendo referencia al artefacto de spring-boot.

La etiqueta *<properties>*, hace referencia a las propiedades que se le dan a la aplicación, en este caso, se le informa que la aplicación correrá con la versión de Java 1.8.La etiqueta *<plugin>*, permite incorporar artefactos empaquetados y listos para utilizar en cualquiera de los módulos de la aplicación.

Iniciando proyecto spring con STS

Iniciemos entonces Spring, cómo está cargado por defecto, observamos en el STS, hay una sección Boot DashBoard, donde nos aparece el título del directorio principal (nombre del artefacto).

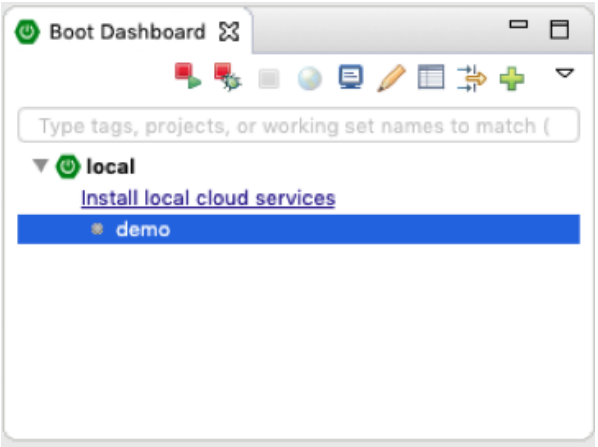

Imagen 12. Boot Dashboard, artefacto.
Fuente: Desafío Latam.

Sobre el nombre, le damos click derecho y a la opción (Re)Start. En la consola de la aplicación, vemos que ya está corriendo la aplicación en base a un servicio de Apache tomcat 9. Entonces, revisamos la información relevante que poseemos aquí:

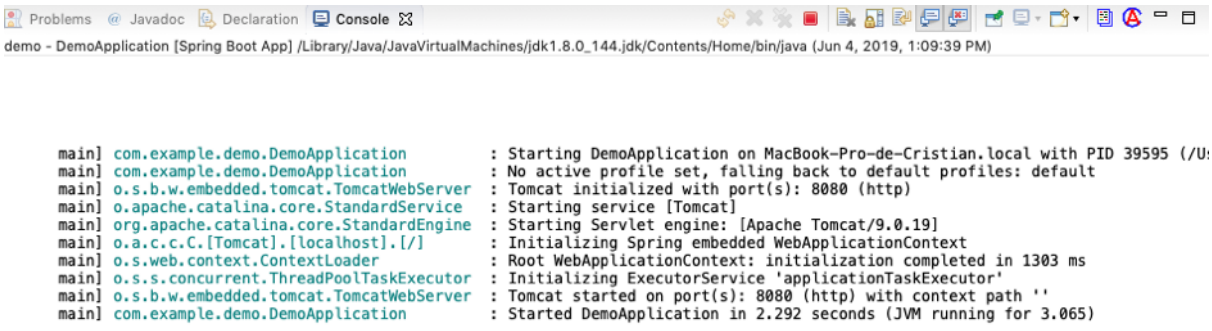

Imagen 13. Información al levantar el servicio.
Fuente: Desafío Latam.

Comienza un servicio Tomcat, en el puerto 8080 (por defecto), inicia Spring embebido con la extensión de aplicación web y otro tipo de información para el conocimiento.

Si vamos a nuestro navegador y entramos a http://localhost:8080/ podremos acceder, ya que el servidor está corriendo para recibir las peticiones HTML, pero nos arrojará un

mensaje de error que la página está en blanco, ya que no hay nada que haga conexión aún con el servicio de spring y el servidor.

Imagen 14. Accediendo a la aplicación.
Fuente: Desafío Latam.

Con esto, demostramos que nuestro sistema spring ya está en ejecución. Iniciando spring de cualquiera de las maneras mencionadas anteriormente, la aplicación funcionará de la misma forma.

Incorporaremos un archivo HTML para mostrar una página estática. Para lograr esto, debemos crear un archivo con el nombre *index.html* en el directorio *src/main/resources/static*, Spring web, se encarga de buscar en los repositorios de la aplicación, y este reconoce automáticamente el archivo, por lo que si lo encuentra, lo muestra en el navegador.

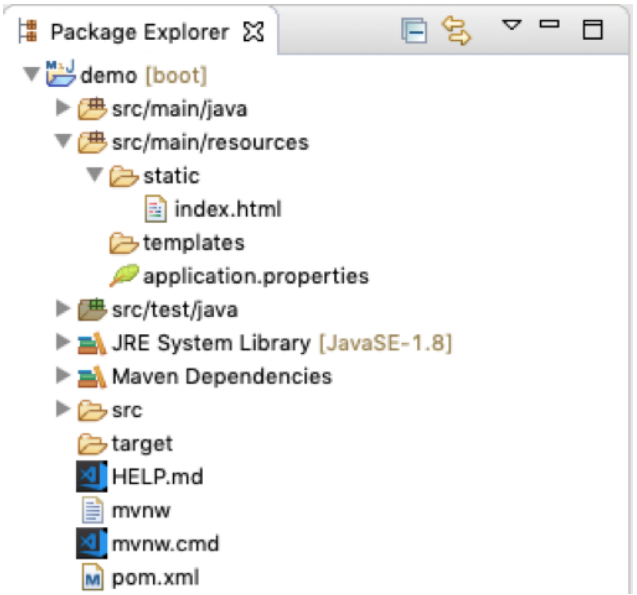

Imagen 15. Agregando el archivo index.html.
Fuente: Desafío Latam.

Nuestro archivo index al poseer contenido estático y será de ejemplo, solo agregaremos un título.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title></title>
</head>
<body>
   <h1>Página de Inicio</h1>
</body>
</html>
```

Ahora, reiniciamos el programa y vemos que ocurre en el navegador, (http://localhost:8080/) efectivamente, ahora al reconocer un archivo de index.html, este es mostrado:

Imagen 16. Mostrando index.html.
Fuente: Desafío Latam.

Si se modifica el contenido estático de una página donde no es afectada por algún funcionamiento de la aplicación, no es necesario que sea reiniciada ahora, si se agregaron nuevas funcionalidades que permiten generar contenido dinámico, ahí sí debe ser reiniciada la aplicación.