SPRING BOOT ENTORNO DE DESARROLLO

GUÍA DE INICIO RÁPIDO

Lo que construirás: Crearás un endpoint clásico de **"¡Hola Mundo!"** al que cualquier navegador podrá conectarse. Incluso puedes decirle tu nombre y responderá de forma más intuitiva.

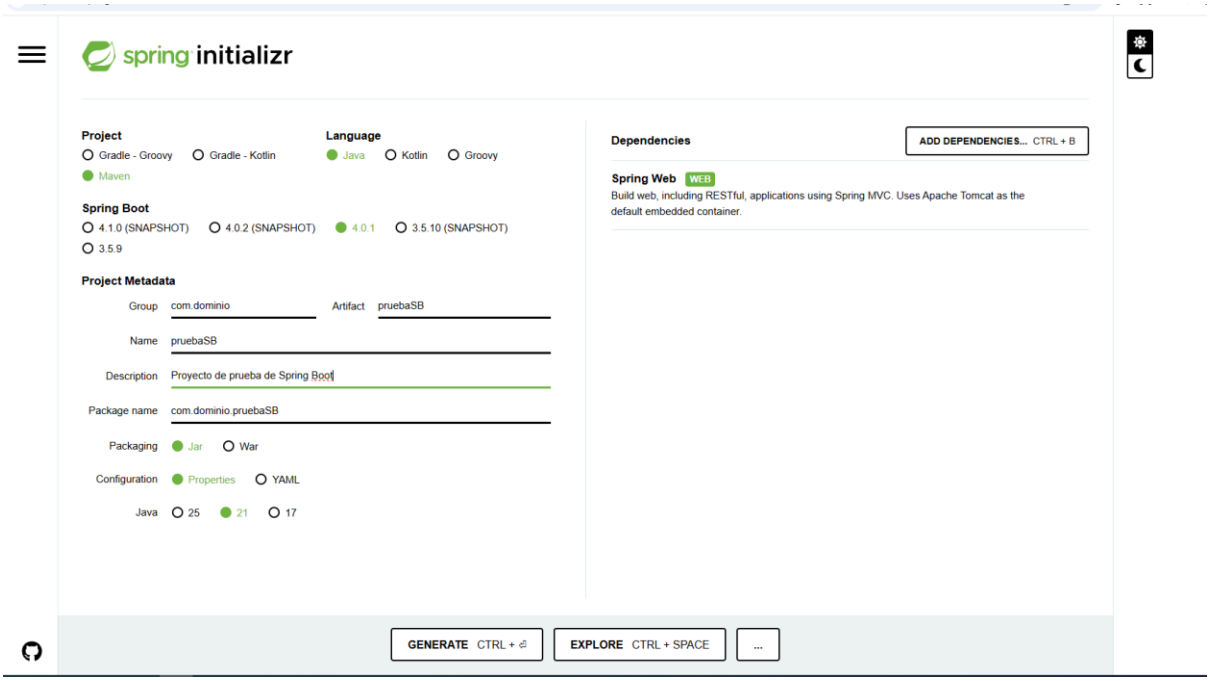
Lo que necesitarás:

- Un entorno de desarrollo integrado (IDE) Las opciones populares incluyen **IntelliJ** IDEA, **Visual Studio Code** con Paquete de extensión Spring Boot, **Eclipse** conHerramientas de resorte O Netbeans.

- Un **kit de desarrollo de Java**™ (JDK) RecomendamosJDK de BellSoft Libericaversión 17 o 21.

Paso 1: Iniciar un nuevo proyecto Spring Boot

Usar start.spring.io Para crear un proyecto web, en el cuadro de diálogo "Dependencias", busque y agregue la dependencia "web" como se muestra en la captura de pantalla. Presione el botón "Generar", descargue el archivo zip y descomprímalo en una carpeta de su computadora.

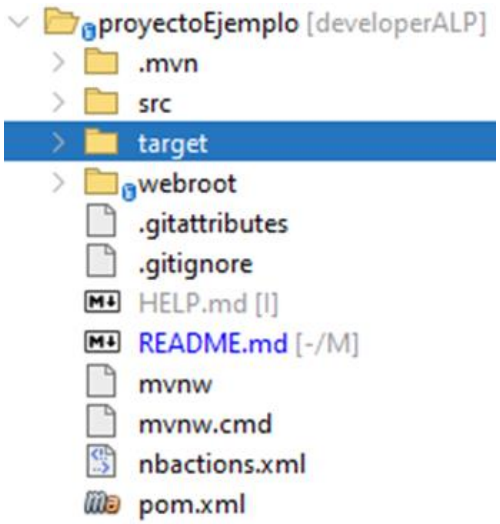


Proyectos creados por start.spring.io contienen un framework Spring Boot que prepara Spring para funcionar en tu aplicación, sin necesidad de mucho código ni configuración. Spring Boot es la forma más rápida y popular de iniciar proyectos Spring.

Paso 2: Estructura de directorios de un proyecto

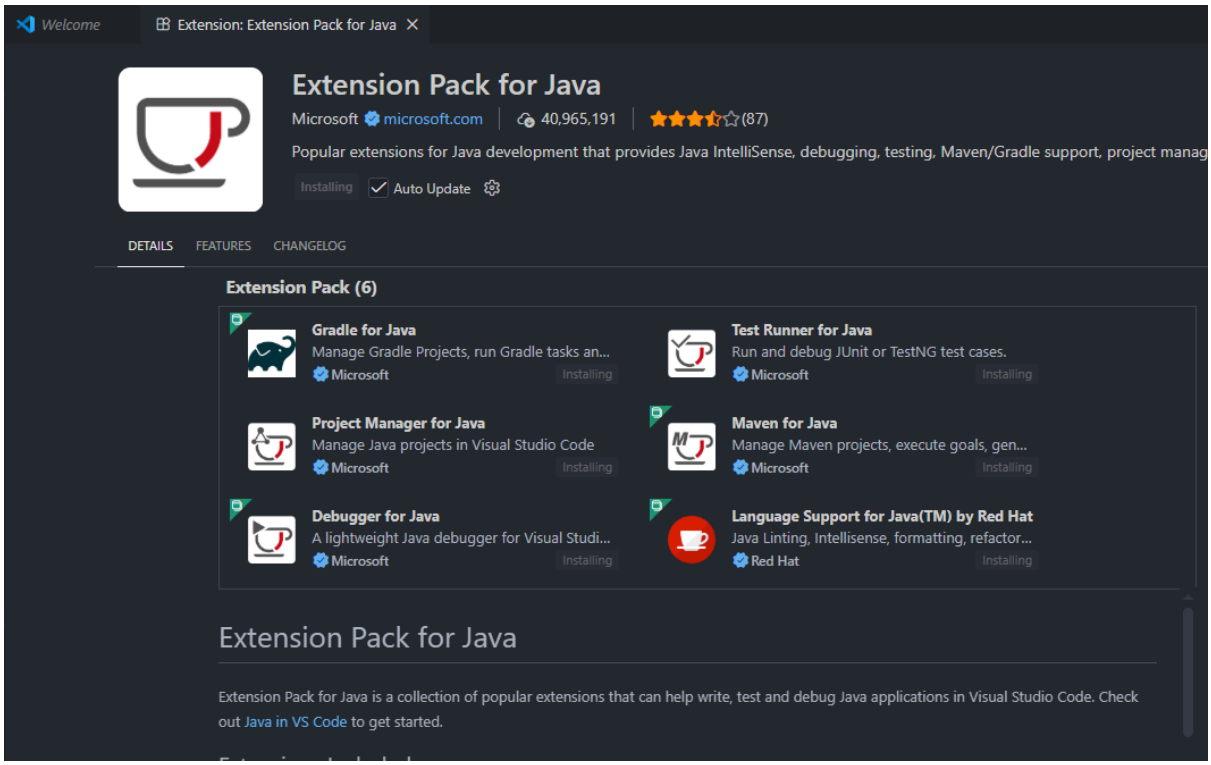
Al crear el proyecto se nos descarga un archivo comprimido con todas las carpetas del proyecto organizadas de la siguiente manera:

- **/.mvn:** carpeta interna de Maven Wrapper en la que están los archivos que usan Maven sin tener que instalarlo globlalmente y garantizar que todos los desarrolladores tengan la misma versión.
- **/src/main/java:** es la carpeta más importante ya que contiene el código real de la aplicación: clases, servicios, repositorios y clase principal.
- **/src/main/resources:** recursos que no son código java como: application.properties o application.yml, archivos de configuración, plantillas y archivos estáticos.
- **/src/main/webapp:** propia de aplicaciones web tradicionales donde se situan: JSP, HTML, CSS y JS. En SpringBoot no siempre se usa.
- **/src/test:** código de tests: unitarios o de integración.
- **/target:** carpeta generada al compilar. Nunca se edita a mano.
- **/webroot:** no es creada por SpringBoot sino por el usuario para guardar el contenido multimedia, estilos y demás archivos manejados por el desarrollador.
- **pom.xml:** el archivo más importante después del código donde se define: dependencias, versión de java usada, plugins y tipo de empaquetado.
- **mvnw y mvnw.cmd:** script de Maven Wrapper para Linux/macOS (el primero), y Windows (el segundo).



Paso 3: Requisitos del entorno

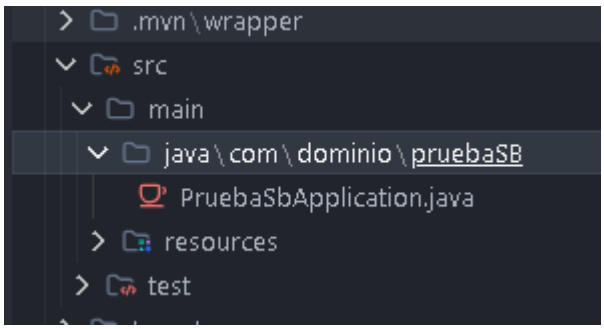
- Nuestro entorno va a ser Windows
- Se necesita tener instalado el JDK, por lo menos la versión 21.
- Los IDE utilizados serán Visual Studio Code y Netbeans.
- Para **Visual Studio** Code se necesitará la extensión de Extension pack for Java



- Se va a utilizar el servidor Apache Tomcat pero ya viene incluido en el proyecto que hemos creado con la página start.spring.io

Paso 4: Agregar el código propio con Visual Studio Code

Abra el proyecto en el IDE y localice el archivo `DemoApplication.java` en la carpeta `src/main/java/com/example/demo` . Ahora modifique el contenido del archivo añadiendo el método y las anotaciones adicionales que se muestran en el código a continuación. Puede copiar y pegar el código o simplemente escribirlo.




```
package com.example.demo;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@RestController
public class DemoApplication {
    public static void main(String[] args) {
      SpringApplication.run(DemoApplication.class, args);
    }
    @GetMapping("/hello")
    public String hello(@RequestParam(value = "name", defaultValue =
"World") String name) {
      return String.format("Hello %s!", name);
    }

}
```

Este es todo el código necesario para crear un servicio web simple “Hola mundo” en Spring Boot.

El método `hello()` que añadimos está diseñado para tomar un parámetro de cadena llamado nombre y combinarlo con la palabra `"Hello"` en el código. Esto significa que si se especifica el nombre `"Amy"` en la solicitud, la respuesta sería "Hola Amy".

La anotación `@RestController` indica a Spring que este código describe un punto final que debe estar disponible en la web. `@GetMapping(“/hello”)` Indica a Spring que use nuestro método `hello()` para responder a las solicitudes enviadas a la dirección `http://localhost:8080/hello`. Finalmente, `@RequestParam` indica a Spring que espere un valor de nombre en la solicitud; si no está presente, usará la palabra `"World"` por defecto.

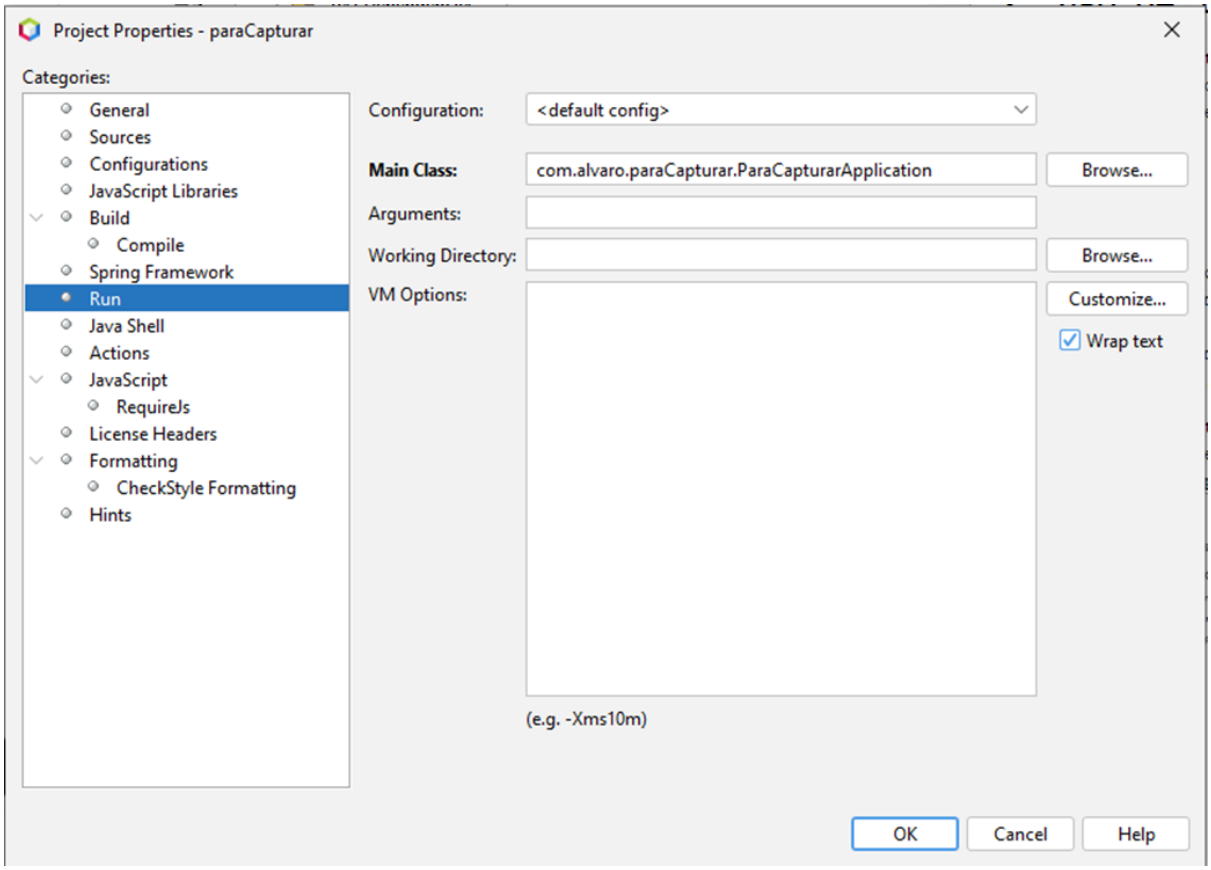
Configurar y editar el proyecto desde NetBeans

En este apartado trataremos los posibles errores que surgen a la hora de iniciar el proyecto <en NetBeans y como se solucionan.

4.1 Error de configuración de proyecto

Al intentar arrancar el proyecto, a veces puede ocurrir que tengamos una mala configuración de NetBeans
(sobretodo si usamos otro lenguaje de programación diferente a Java).

Para solventar este problema debemos ir a Properties>Run y en el apartado Main Class seleccionamos el paquete correspondiente.



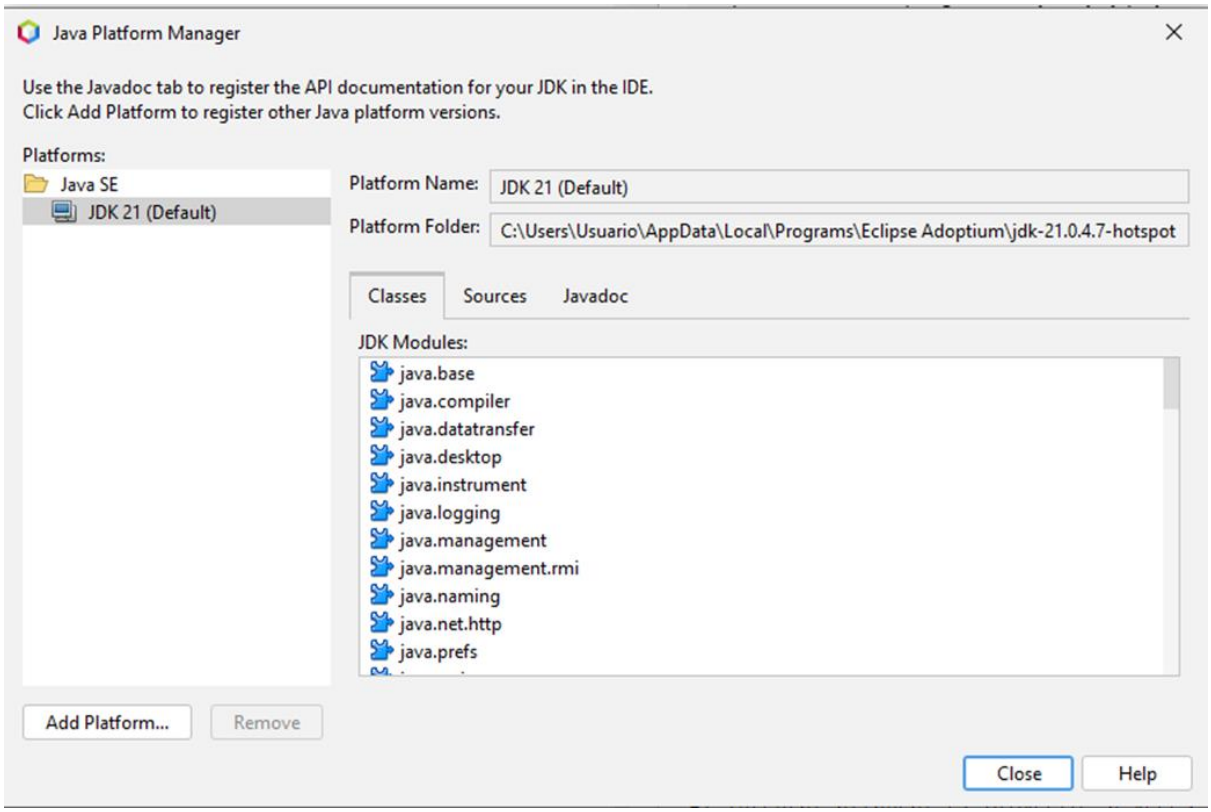
Una vez realizado realizamos un Clean del proyecto y ejecutamos el main.

4.2 Versión de JDK incompatible

En caso de tener una versión del JDK distinta a la de SpringBoot tenemos dos opciones:
- Crear un nuevo proyecto con la versión del JDK que tenemos descargada. - Descargar la versión del JDK correspondiente y/o en caso de tenerlo, configurar el NetBeans.

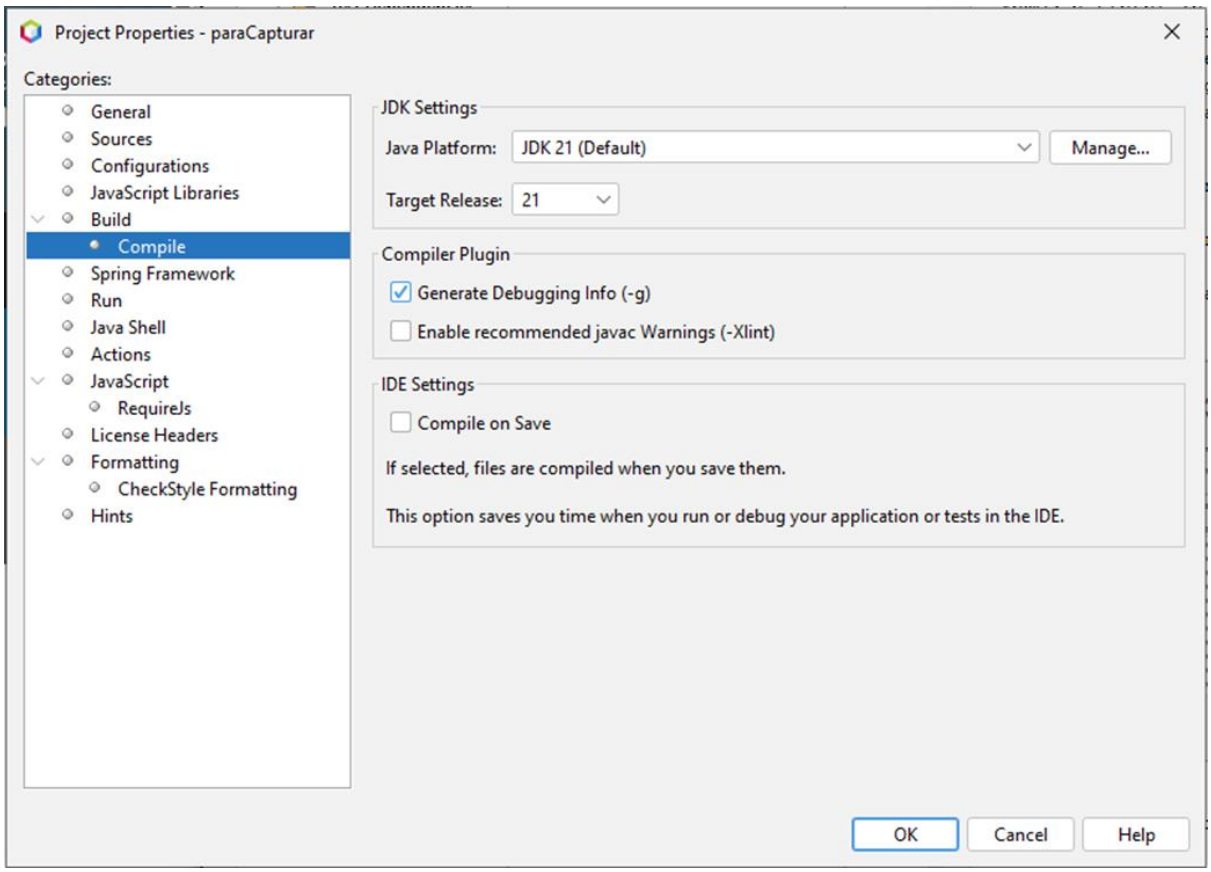
Vamos a tratar la segunda opción. Para descargar el JDK debemos ir al sitio oficial:
https://www.oracle.com/java/technologies/downloads/
En este caso escogemos la versión 21 y guardamos los archivos en una carpeta al gusto. Importante guardar la ruta.

Abrimos el NetBeans y vamos al apartado Tools>Java Platforms
Se abrirá un cuadro en el que aparecen todas las versiones del JDK tenemos instaladas y nos dará la posibilidad de añadir más.



En caso de que no aparezca la versión que queremos (en este caso, la 21) debemos añadirla.

Una vez añadido y comprobado que está la versión correcta, nos dirigimos a Properties>Build>Compile
y comprobamos que el Java Platform tiene la versión correcta:



Realizamos un Clean al proyecto y ejecutamos el main para probar que funciona.

Paso 5: Pruébalo

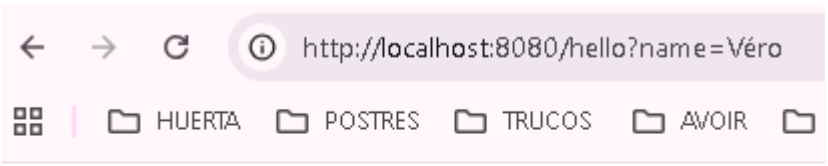
Construyamos y ejecutemos el programa. Abra una línea de comandos (o terminal) y navegue a la carpeta donde se encuentran los archivos del proyecto. Podemos compilar y ejecutar la aplicación con el siguiente comando:

En windows:

```
.\gradlew.bat bootRun
```

Deberías ver un resultado muy similar a esto:

Las últimas líneas indican que Spring ha comenzado. El servidor Apache Tomcat integrado de Spring Boot actúa como servidor web y escucha las solicitudes en el puerto `localhost:8080`. Abra su navegador y escriba `http://localhost:8080/hello?name=Nombre`. Debería obtener una respuesta amigable como esta:



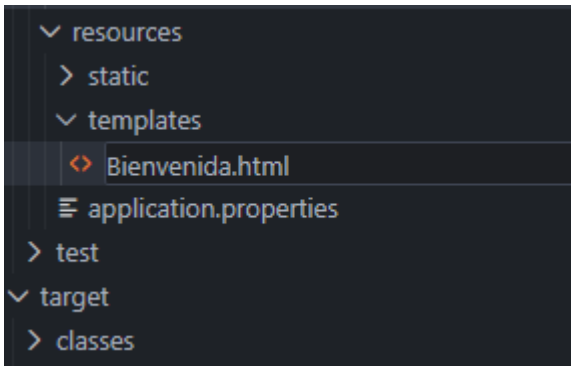
Hello Véro!

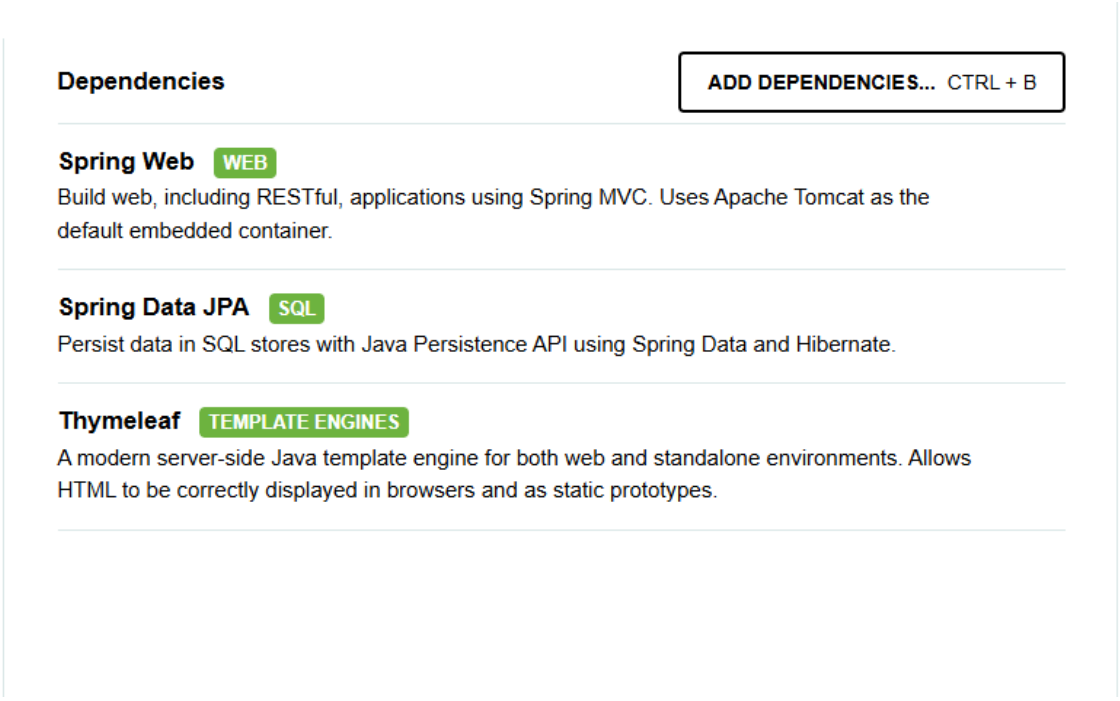
Trabajando con Spring Boot

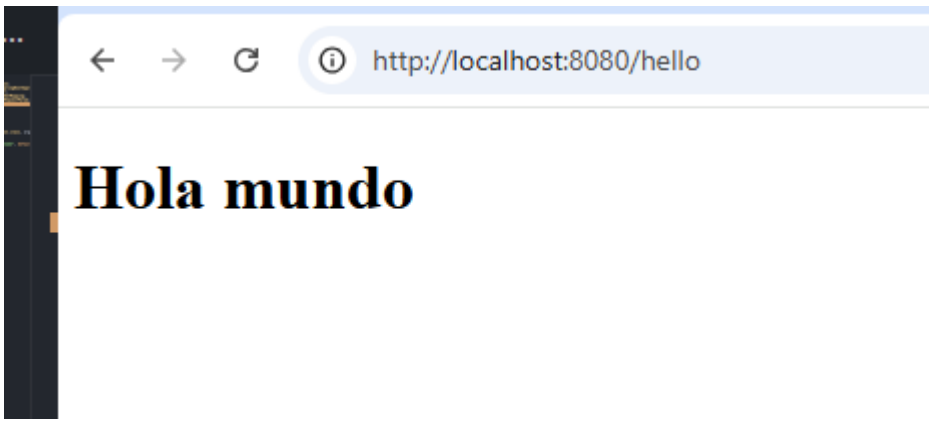
Versionar con GITHUB

Subir a Explotación

SE crean los html en resources-templates







PruebaSbApplic

Bienvenida.html

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Bienvenida-SpringBoot</title>
</head>
<body>
    <h1>Hola mundo</h1>
    <p th:text="'Bienvenid@ '+ ${nombre}"></p>
</body>
</html>
```

pagina2.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org
<head>
    <title>Document</title>
</head>
<body>
    <h1>¡Hola desde Thymeleaf!</h1>
    <p th:text="'Bienvenido a la ' + ${n
    <a href="/">Volver</a>
</body>
</html>
```