Spring Boot - Hello World

Last Updated : 23 Jul, 2025

features of Spring. It has become a favorite of developers these days because of its rapid production-ready environment, which enables the developers to directly focus on the logic instead of struggling with the configuration and setup. Spring Boot is a microservice-based framework and making a production-ready application in it takes very little time. In this article, we will **print "Hello World" using Spring Boot.**

This article demonstrates two ways to print "Hello World" using Spring Boot:

1. Using the CommandLineRunner interface in Spring Boot
2. Using a Controller Class in Spring Boot

First, initialize the project on our machine. Spring Initializr is a web-based tool that we can use to easily generate the structure of the Spring Boot project. It also provides different features for the projects expressed in a metadata model. This model allows us to configure the list of dependencies that are supported by the JVM. Here, we will create the structure of an application using a spring initializer and then use an IDE to create a sample GET route. Therefore, to do this, the following steps are followed sequentially.

Step-by-Step Implementation

Step 1: Initialize the Spring Boot Project

We will use **Spring Initializr** to generate the Spring Boot project structure.

Go to Spring Initializr. Fill in the details as per the requirements.

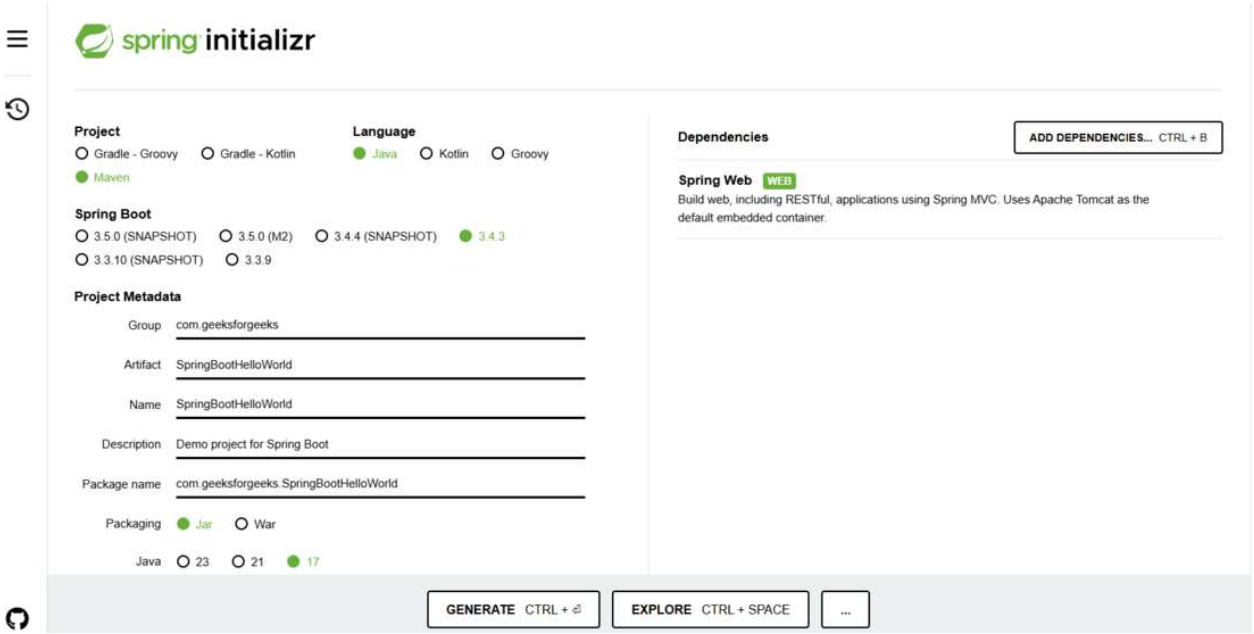
***Project**: Maven*
***Language**: Java*
***Spring Boot:** 3.4.3 (or the latest stable version)*
***Packaging:** JAR*
***Java:** 17 (or later)*
***Dependencies:** Spring Web*

Here, we are using STS IDE

Provide the following details:

- **Group**: com.geeksforgeeks
- **Artifact**: SpringBootHelloWorld

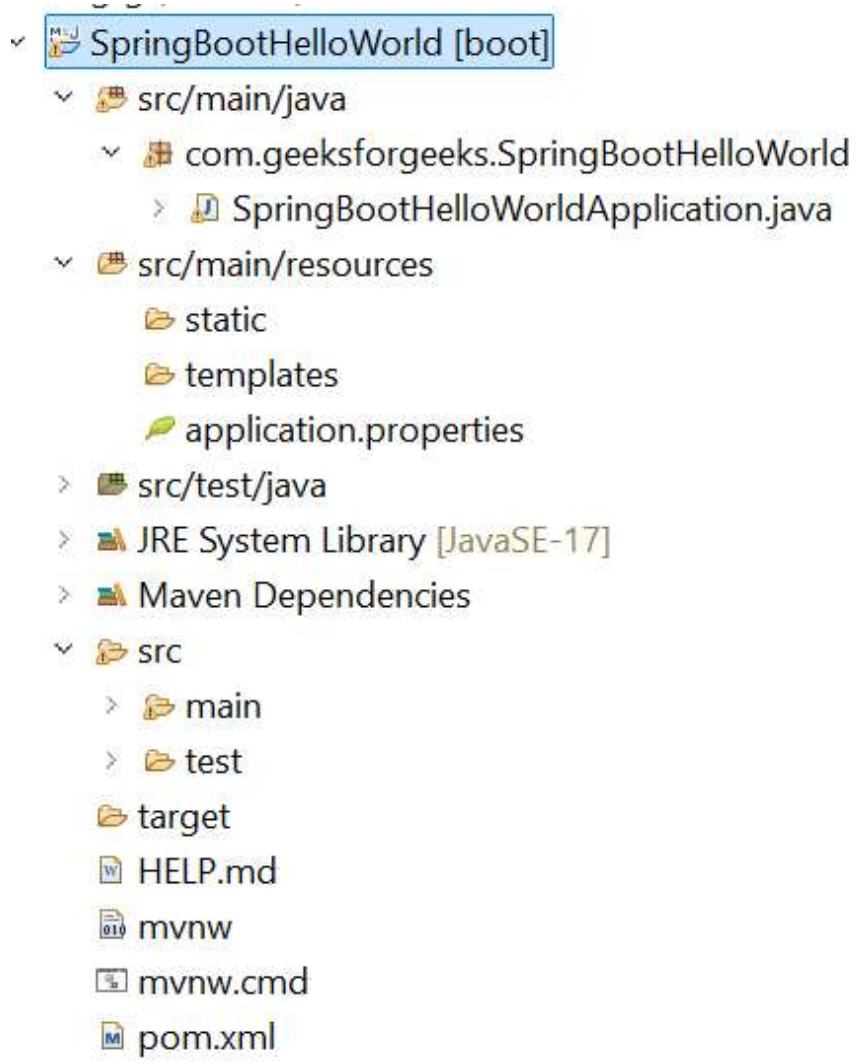
Click on "Generate" to download the project as a ZIP file. Then Extract the ZIP file to a suitable location on your system.



Step 2: Import the Project in an IDE

After extract the zip file, now open STS IDE and then go to **File > Import> Existing Maven Project > Next > Browse > Select the Project Folder > Finish**.

After the project imports successfully, it will look like pictorially depicted as below:



Note: In the Import Project for Maven window, make sure you choose the same version of JDK which you selected while creating the project.

Method 1: Using CommandLineRunner Interface

In this method, we will use the CommandLineRunner interface to print "Hello world" when the application starts.

Step 3: Create the Main Application Class

- Go to src > main > java > com.geeksforgeeks.
- Open the SpringBootHelloWorldApplication.java file.

Replace its content with the following code:

```
package com.geeksforgeeks.SpringBootHelloWorld;

import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

//Main class
//Implementing CommandLineRunner interface
public class SpringBootHelloWorldApplication
 implements CommandLineRunner {

 // Method 1
 public void run(String args[]) throws Exception
 {
     // Print statement when method is called
     System.out.println("HEllo world");
 }

 // Method 2
 // Main driver method
 public static void main(String[] args)
 {
     // Calling run() method to execute
     // SpringBootApplication by
     // invoking run() inside main() method
     SpringApplication.run(
         SpringBootHelloWorldApplication.class, args);
 }

}
```

This application is now ready to run.

Step 4: Run the Spring Boot Application

Run the SpringBootHelloWorldApplication class and wait for the Tomcat server to start where the default port is already set.

> *Tip: The default port of the Tomcat server is 8080 and can be changed in the application.properties file.*

Output: Generated on terminal/CMD

```
2021-10-26 10:21:31.027  INFO 24312 --- [           main] JPA.demo.DemoApplication
HEllo world
```

Method 2: Using a Controller Class

In this method, we will create a controller class to handle incoming HTTP requests and return "Hello World" as a response.

Step 5: Create a Controller Class

Go to **src > main > java > com.geeksforgeeks** and create a new Java class named **HelloWorldController.java**. Below is the code for the controller.java file.

```java
package com.geeksforgeeks.SpringBootHelloWorld.controller;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

// Marking this class as a REST controller
@RestController
public class HelloWorldController {

    // Mapping the root URL ("/") to this method
    @RequestMapping("/")
    public String helloWorld() {

        // Returning a simple "Hello World" response
        return "Hello World";
    }
}
```

This controller helps to handle all the incoming requests from the client-side.

Step 6: Run the Application

- Run the Spring Boot application inside your IDE.
- Wait for the Tomcat server to start (Default port: 8080).

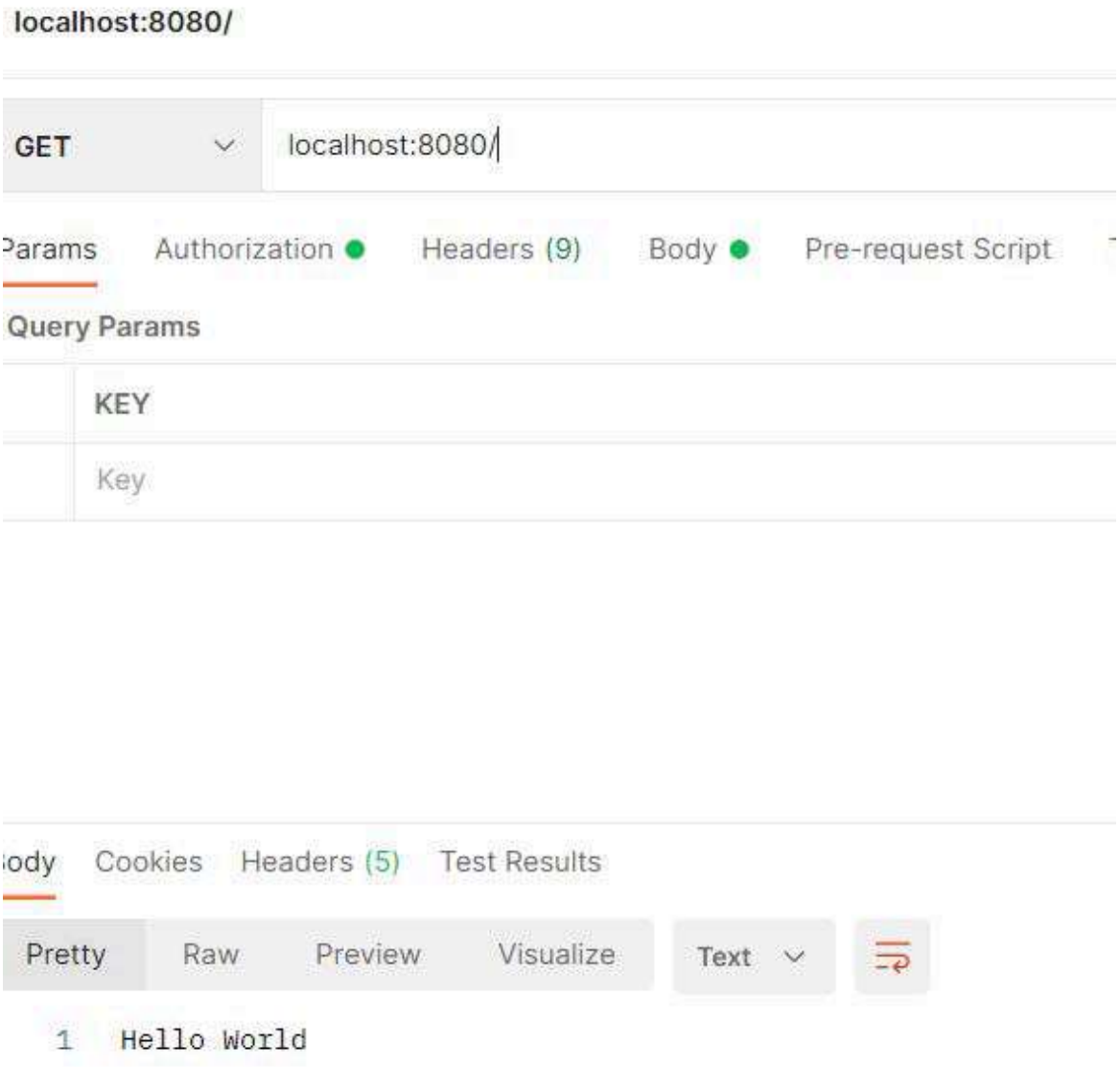
When the application runs successfully, it will show the the message that the application has started in the default port number 8080 as shown in following image:

```
security configuration must be updated before running your application in production.

main] o.s.s.web.DefaultSecurityFilterChain     : Will secure any request with [org.springframework.security.web.session.DisableEncodeUr
main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
main] c.g.SpringBootHelloWorldApplication      : Started SpringBootHelloWorldApplication in 6.197 seconds (process running for 6.925)
```

Now we will use the PostMan and call the get API of the Spring boot application.

Visit **http://localhost:8080/ in postman with GET request:**



Comment | More info | Advertise with us