

[< ALL GUIDES](#)

Spring Quickstart Guide

What you'll build

You will build a classic “Hello World!” endpoint which any browser can connect to. You can even tell it your name, and it will respond in a more friendly way.

What you'll need

An Integrated Developer Environment (IDE)

Popular choices include [IntelliJ IDEA](#), [Spring Tools](#), [Visual Studio Code](#), or [Eclipse](#), and many more.

A Java™ Development Kit (JDK)

We recommend [AdoptOpenJDK](#) version 8 or version 11.

Step 1: Start a new Spring Boot project

Use start.spring.io to create a “web” project. In the “Dependencies” dialog search for and add the “web” dependency as shown in the screenshot. Hit the “Generate” button, download the zip, and unpack it into a folder on your computer.

The current version of Spring Boot changes regularly. Just choose the latest release (but not snapshot).

Click on 'Add dependencies', type 'Web' in the search box, then click on the dependency 'Spring Web' to select it.

The Spring Initializr interface shows the following configuration:

- Project:** ☒ Maven Project, ☐ Gradle Project
- Language:** ☒ Java, ☐ Kotlin, ☐ Groovy
- Spring Boot:** ☐ 2.3.0 M3, ☐ 2.3.0 (SNAPSHOT), ☐ 2.2.7 (SNAPSHOT), ☒ 2.2.6, ☐ 2.1.14 (SNAPSHOT), ☐ 2.1.13
- Project Metadata:**
 - Group: com.example
 - Artifact: demo
 - Name: demo
 - Description: Demo project for Spring Boot
 - Package name: com.example.demo
- Packaging:** ☒ Jar, ☐ War
- Java:** ☐ 14, ☐ 11, ☒ 8
- Dependencies:** ☒ Spring Web (WEB). Description: Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Buttons at the bottom: GENERATE, EXPLORE, CTRL + SPACE, SHARE...

Projects created by start.spring.io contain [Spring Boot](#), a framework that makes Spring ready to work inside your app, but without much code or configuration required. Spring Boot is the quickest and most popular way to start Spring projects.

Step 2: Add your code

Open up the project in your IDE and locate the `DemoApplication.java` file in the `src/main/java/com/example/demo` folder. Now change the contents of the file by adding the extra method and annotations shown in the code below. You can copy and paste the code or just type it.

```
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);
    }
}
```

[COPY](#)

```
@GetMapping("/hello")
public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
    return String.format("Hello %s!", name);
}
```

This is all the code required to create a simple “Hello World” web service in Spring Boot.

The `hello()` method we’ve added is designed to take a String parameter called `name`, and then combine this parameter with the word `"Hello"` in the code. This means that if you set your name to `"Amy"` in the request, the response would be `"Hello Amy"`.

The `@RestController` annotation tells Spring that this code describes an endpoint that should be made available over the web. The `@GetMapping("/hello")` tells Spring to use our `hello()` method to answer requests that get sent to the `http://localhost:8080/hello` address. Finally, the `@RequestParam` is telling Spring to expect a `name` value in the request, but if it’s not there, it will use the word “World” by default.

Step 3: Try it

Let’s build and run the program. Open a command line (or terminal) and navigate to the folder where you have the project files. We can build and run the application by issuing the following command:

MacOS/Linux:

```
./mvnw spring-boot:run
```

Windows:

```
mvnw spring-boot:run
```

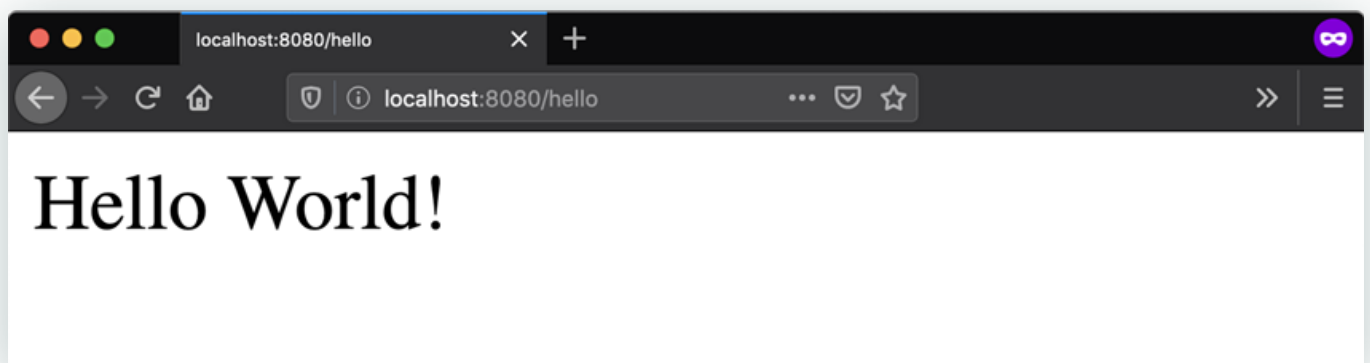
You should see some output that looks very similar to this:

```
demo ./mvnw spring-boot:run --quiet

:: Spring Boot :: (v2.2.4.RELEASE)

2020-02-14 16:16:47.746 INFO 4838 --- [main] com.example.demo.DemoApplication : Starting DemoApplication
on Brians-MacBook-Pro.local with PID 4838 (/Users/bclozel/workspace/tmp/demo/target/classes started by bclozel in /Users/bclozel/workspace/tmp/demo)
2020-02-14 16:16:47.748 INFO 4838 --- [main] com.example.demo.DemoApplication : No active profile set, falling back to default profiles: default
2020-02-14 16:16:48.272 INFO 4838 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2020-02-14 16:16:48.279 INFO 4838 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2020-02-14 16:16:48.279 INFO 4838 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.30]
2020-02-14 16:16:48.323 INFO 4838 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2020-02-14 16:16:48.324 INFO 4838 --- [main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 532 ms
2020-02-14 16:16:48.438 INFO 4838 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2020-02-14 16:16:48.533 INFO 4838 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s) 8080 (http) with context path ''
2020-02-14 16:16:48.535 INFO 4838 --- [main] com.example.demo.DemoApplication : Started DemoApplication in 1.006 seconds (JVM running for 1.248)
```

The last couple of lines here tell us that Spring has started. Spring Boot's embedded Apache Tomcat server is acting as a webserver and is listening for requests on `localhost` port `8080`. Open your browser and in the address bar at the top enter `http://localhost:8080/hello`. You should get a nice friendly response like this:



? Pop quiz

What should happen if you add `?name=Amy` to the end of the URL?

Next, try these popular guides

You've already seen how simple Spring can be, but it's also very flexible. There are thousands of things you can do with Spring, and we have lots of guides available to take you through the most popular choices. Why not keep on learning and try one of these additional guides?



Building a RESTful web Service

Continue your learning by creating a RESTful JSON web service in Spring



Consuming a RESTful Web Service

Learn how to retrieve web page data with Spring's RestTemplate.



Accessing Data with JPA

Learn how to work with JPA data persistence using Spring Data JPA.