

Spring Boot Simple Hello World

Description: Spring Boot is an open source, micro service-based Java web framework. The Spring Boot framework creates a fully production-ready environment that is completely configurable using its prebuilt code within its own codebase.

Project: 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. This project is based on [Spring Quickstart Guide](#) and just expands on the screenshots and step by step instructions.

Technology: This project uses the following technology:

Integrated Development Environment (IDE):

[Spring Tool Suite 4](#) (Version: 4.11.0.RELEASE)

[Visual Studio Code](#) (V8: 9.8.177.11)

Java Development Kit (JDK):

[Oracle's JDK 8](#) (1.8)

Table of Contents

Glossary of Terminology	2
Generate Spring Boot Download	2
Import the Spring Boot Download	2
Import the project: “demo-spring-boot-hello”	2
Create the Hello World Project from the Import.....	3
Create an API GET Endpoint.....	3
Add RestController class annotation	3
Add GetMapping Annotation and Endpoint Method	4
Copy and paste source code	5
Project Changes Explained: DemoSpringBootTestApplication	6
Class annotation: <code>@RestController</code>	6
Method annotation <code>@GetMapping()</code> API: /hello	6
Method hello()	6
Run the Project	7

Glossary of Terminology

For a list of key terms and definitions used throughout this and various Spring Boot demo documents see the document titled “Appendix 01 Glossary”.

Generate Spring Boot Download

Follow the instructions in the document title “Appendix 02 Spring Initializr” to generate a spring boot download for this project. When the document talks about the name of the project the “**Artifact**” name use “**demo-spring-boot-hello**”.

Import the Spring Boot Download

Import the project: “demo-spring-boot-hello”

Follow the instructions in the document title “Appendix 03 Import Project” and import the “**demo-spring-boot-hello**” project that was created with Spring Initializr.

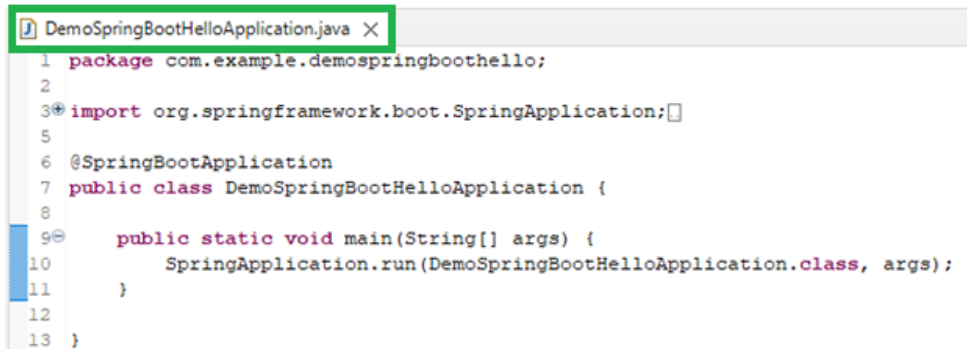
Create the Hello World Project from the Import

Open the Spring Tool Suite 4 IDE.

Create an API GET Endpoint

We will make the main class a rest controller and add a get mapping API.

Open and update class: DemoSpringBootHelloApplication

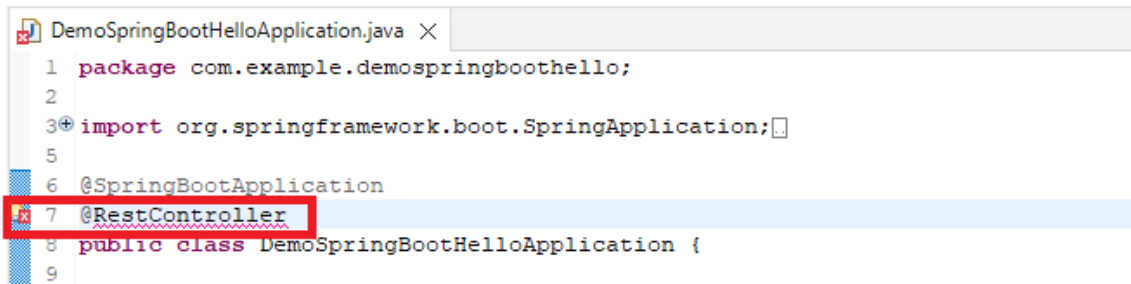


```
1 package com.example.demospringboothello;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6 public class DemoSpringBootHelloApplication {
7
8     public static void main(String[] args) {
9         SpringApplication.run(DemoSpringBootHelloApplication.class, args);
10    }
11 }
12
13 }
```

Add RestController class annotation

RestController - Used for making restful web services. This annotation is used at the class level and allows the class to handle the requests made by the client.

`@RestController`



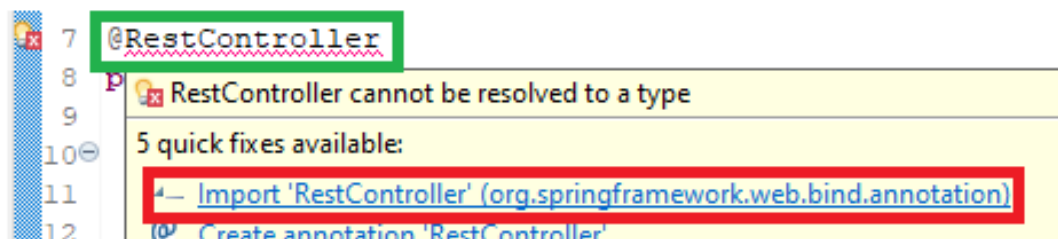
```
1 package com.example.demospringboothello;
2
3 import org.springframework.boot.SpringApplication;
4
5 @SpringBootApplication
6
7 @RestController
8 public class DemoSpringBootHelloApplication {
9 }
```

Correct the import error:

Use the IDE wizard to help correct the error.

Hover the mouse over the error.

Select the import link shown below.



```
7 @RestController
8
9
10
11
12
```

RestController cannot be resolved to a type

5 quick fixes available:

- Import 'RestController' (org.springframework.web.bind.annotation)
- Create annotation 'RestController'

The RestController import gets added clearing the error.

```
DemoSpringBootHelloApplication.java X
1 package com.example.demospringboothello;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.web.bind.annotation.RestController;
6
7 @SpringBootApplication
8 @RestController
9 public class DemoSpringBootHelloApplication {
```

Add GetMapping Annotation and Endpoint Method

GetMapping maps HTTP GET requests onto specific handler methods.

Add Endpoint Method: hello

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

```
14
15 @GetMapping("/hello")
16 public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
17     return String.format("Hello %s!", name);
18 }
```

Resolve the import errors:

Use the IDE wizard to help resolve the errors.

Hover the mouse over the error.

Select the import link shown below.

```
14
15 @GetMapping("/hello")
16 public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
17     return String.format("Hello %s!", name);
18 }
19
```

GetMapping cannot be resolved to a type

9 quick fixes available:

Import 'GetMapping' (org.springframework.web.bind.annotation)

```
16 @GetMapping("/hello")
17 public String hello(@RequestParam value = "name", defaultValue = "World") String name) {
18     return String.format("Hello %s!", name);
19 }
20
21 }
```

RequestParam cannot be resolved to a type

6 quick fixes available:

Import 'RequestParam' (org.springframework.web.bind.annotation)

The imports get added clearing the errors.

```
DemoSpringBootHelloApplication.java ×
1 package com.example.demospringboothello;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestParam;
7 import org.springframework.web.bind.annotation.RestController;

17 @GetMapping("/hello")
18 public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
19     return String.format("Hello %s!", name);
20 }
```

Copy and paste source code

As an alternative you can copy the code below and replace all code in the class as long as you used the same package name.

```
package com.example.demospringboothello;

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 DemoSpringBootHelloApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoSpringBootHelloApplication.class, args);
    }

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

```
DemoSpringBootHelloApplication.java ×
1 package com.example.demospringboothello;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.web.bind.annotation.GetMapping;
6 import org.springframework.web.bind.annotation.RequestParam;
7 import org.springframework.web.bind.annotation.RestController;
8
9 @SpringBootApplication
10 @RestController
11 public class DemoSpringBootHelloApplication {
12
13     public static void main(String[] args) {
14         SpringApplication.run(DemoSpringBootHelloApplication.class, args);
15     }
16
17     @GetMapping("/hello")
18     public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
19         return String.format("Hello %s!", name);
20     }
21
22 }
```

Project Changes Explained: DemoSpringBootHelloApplication

Class annotation: `@RestController`

```
10 @RestController
11 public class DemoSpringBootHelloApplication {
```

The `@RestController` annotation tells Spring that this code describes an endpoint that should be made available over the web.

Method annotation `@GetMapping()` API: `/hello`

```
17 @GetMapping("/hello")
18 public String hello(@RequestParam(value = "name", defaultValue = "World") String name)
```

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

Method `hello()`

```
17 @GetMapping("/hello")
18 public String hello(@RequestParam(value = "name", defaultValue = "World") String name) {
19     return String.format("Hello %s!", name);
20 }
```

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".

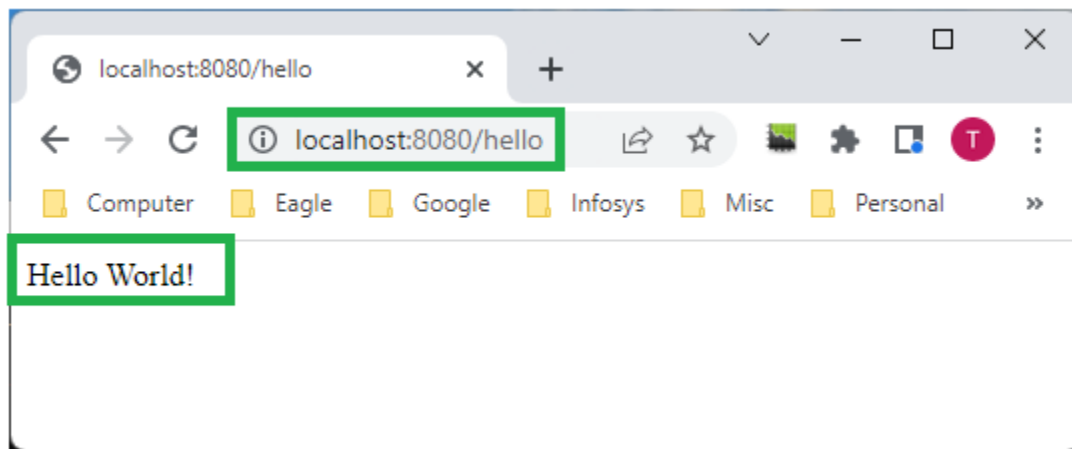
Run the Project

Follow the instructions in the document title “Appendix 04 Run Spring Initializr Project” to test the demo created in this document. The document “Appendix 04 Run Spring Initializr Project” was created based on this demo so no additional test instructions are provided here.

Expected results

The “Appendix 04 Run Spring Initializr Project” will use two URLs to test the project in this document.

<http://localhost:8080/hello>



<http://localhost:8080/hello?name=Tom>

