Lab 2 – Hello World Greeting

To complete this exercise, you must have IntelliJ installed, as well as the latest versions of Java and Maven. Lab 1 must be completed prior to this lab; please do not proceed if you haven't yet done so.

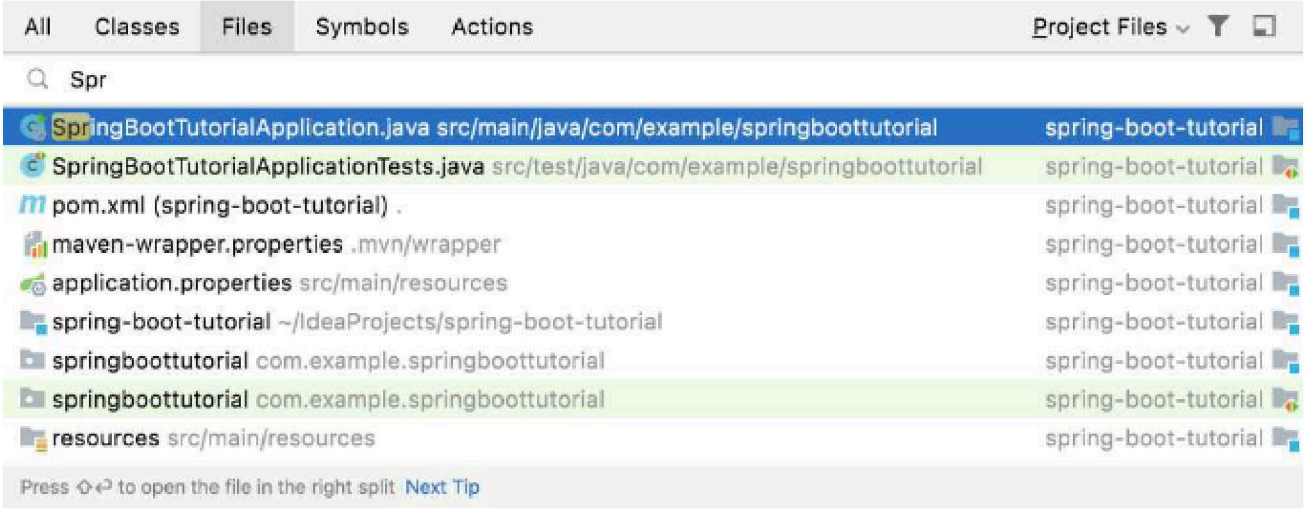
The purpose of this task is to utilise the Spring Web dependency by creating a method which returns a greeting to the user and mapping it to the HTTP endpoint accessed via the browser.

Task 1

Step-by-step guide to add a method to send a greeting:

1. Open the SpringBootTutorialApplication.java file under src/main/java/com/example/springboottutorial.

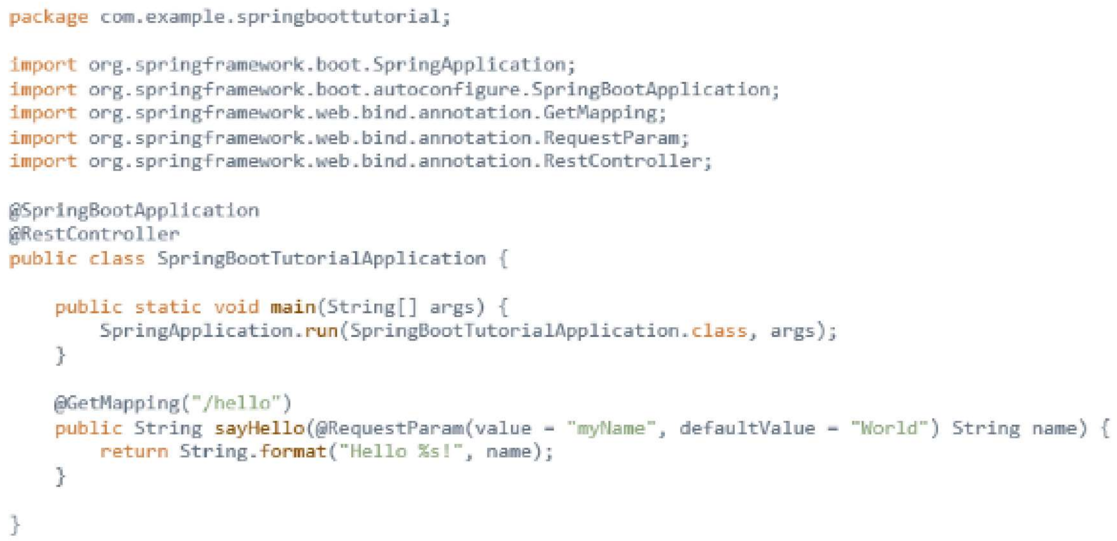
   *Tip: IntelliJ IDEA provides the **Go to File** action to quickly find and open files. From the main menu, select **Navigate -> File** or press **Ctrl+Shift+N**, start typing the name of the file and select it from the list.*



Spr

SpringBootTutorialApplication.java src/main/java/com/example/springboottutorial	spring-boot-tutorial
SpringBootTutorialApplicationTests.java src/test/java/com/example/springboottutorial	spring-boot-tutorial
pom.xml (spring-boot-tutorial) .	spring-boot-tutorial
maven-wrapper.properties .mvn/wrapper	spring-boot-tutorial
application.properties src/main/resources	spring-boot-tutorial
spring-boot-tutorial ~/IdeaProjects/spring-boot-tutorial	spring-boot-tutorial
springboottutorial com.example.springboottutorial	spring-boot-tutorial
springboottutorial com.example.springboottutorial	spring-boot-tutorial
resources src/main/resources	spring-boot-tutorial

2. Add the **sayHello()** method with all of the necessary annotations and imports so that the file looks like the example on the following page:



```
package com.example.springboottutorial;

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 SpringBootTutorialApplication {

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

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

}
```

3. The sayHello() method takes the name parameter and returns the word Hello combined with the parameter value. Everything else is handled by adding Spring annotations:

3.1 The @RestController annotation marks the SpringBootTutorialApplication class as a request handler (a REST controller).

3.2 The @GetMapping("/hello") annotation maps the sayHello() method to GET requests for /hello.

The @RequestParam annotation maps the name method parameter to the myName web request parameter. If you don't provide the myName parameter in your web request, it will default to World.

Task 2 - Run a spring application:

IntelliJ IDEA creates a Spring Boot run configuration that you can use to run your new Spring application. If the run configuration is selected, press **Shift+F10**.

You can also use the **Run** icon in the gutter of the SpringBootTutorialApplication.java file next to the class declaration or the main() method declaration.

By default, IntelliJ IDEA shows your running Spring Boot application in the Run tool window.

The Console tab shows the output of Spring log messages. By default, the built-in Apache Tomcat server is listening on port 8080. Open your web browser and go to http://localhost:8080/hello. If you did everything right, you should see your application respond with Hello World!