

How to Make a Simple RestController in Spring Boot?

Last Updated : 23 Jul, 2025

A **RestController in Spring Boot** is a specialized controller that is used to develop RESTful web services. It is marked with the `@RestController` annotation, which combines `@Controller` and `@ResponseBody`. This ensures that the response is automatically converted into **JSON or XML**, eliminating the need for explicit response conversions. In this article, we will explore **how to create a simple RestController in Spring Boot**.

Steps to Create a Simple RestController in Spring Boot

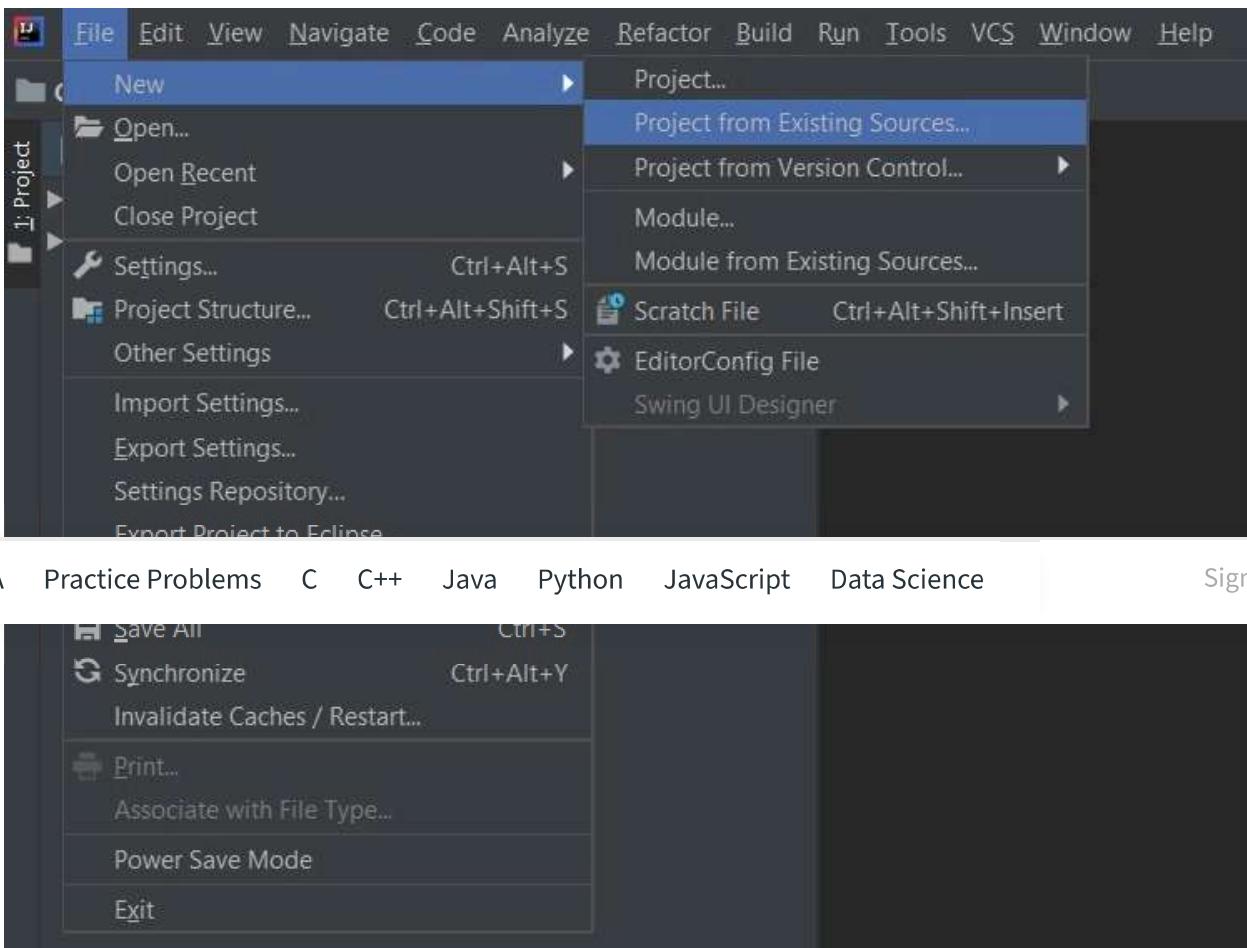
Step 1: Setup Spring Boot Project Using Spring Initializr

- Go to [Spring Initializr](#).
- Configure the project with the following details:
 - **Project:** Maven
 - **Language:** Java
 - **Spring Boot Version:** 3.x (Latest Version)
 - **Packaging:** JAR
 - **Java Version:** 17 or later
 - **Dependencies:** Spring Web
- Click on **Generate** to download the project.

The screenshot shows the Spring Initializr web interface. At the top, there are sections for 'Project' (Gradle - Groovy, Gradle - Kotlin, Maven), 'Language' (Java, Kotlin, Groovy, Java is selected), and 'Dependencies' (Spring Web, WEB). Below these are sections for 'Spring Boot' (3.5.0 (SNAPSHOT), 3.5.0 (M2), 3.4.4 (SNAPSHOT), 3.4.3, 3.3.10 (SNAPSHOT), 3.3.9, 3.4.3 is selected) and 'Project Metadata' (Group: com.geeksforgeeks, Artifact: SpringBootApplication, Name: SpringBootApplication, Description: Demo project for Spring Boot, Package name: com.geeksforgeeks.SpringBootApplication). At the bottom, there are buttons for 'GENERATE' (CTRL + D), 'EXPLORE' (CTRL + SPACE), and an ellipsis (...).

Step 2: Import the Project into Your IDE

Extract the zip file. Now open a suitable IDE and then go to **File > New > Project from existing sources** and select **pom.xml**. Click on import changes on prompt and wait for dependencies to load.



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

Step 3: Create a Simple RestController

Go to **src > main > java > com.geeksforgeeks.SpringBootApplication**, create a new Java class with the name **Controller.java** and add the annotation **@RestController**.

Controller.java:

```
package com.geeksforgeeks.SpringApplication.controller;

import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/api")
public class Controller {

    @GetMapping("/hello/{name}/{age}")
    public String sayHello(@PathVariable String name, @PathVariable int age)
```

```
{
    return "Hello, " + name + "! You are " + age + " years old.";
}
}
```

Explanation:

- `@RestController` annotation marks this class as a RESTful controller.
- `@RequestMapping("/api")` defines a base URL for all endpoints inside this controller.
- `@GetMapping("/hello/{name}/{age}")` handles HTTP GET requests and extracts name and age from the URL.
- `@PathVariable` annotation captures dynamic values from the request URL.
- Returns a simple JSON response with a greeting message.

Step 4: Run the Spring Boot Application

Run the main class and wait for the Tomcat server to start.

```
main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.35]
main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
main] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 1747 ms
main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
main] o.s.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
main] c.g.S.boot.app.SpringBootAppApplication : Started SpringBootAppApplication in 2.882 seconds (JVM running for 3.514)
```

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

This **Controller.java** file is used for handling all incoming requests from the client-side.

Once the application starts, test the API by opening a browser or using [Postman](#):

`http://localhost:8080/api/hello/Sweta/25`

Response:

"Hello, Sweta! You are 25 years old."