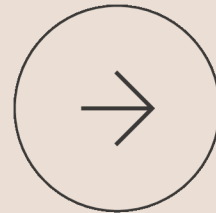


Building RESTful Web Services

using Spring Boot

Part - 1

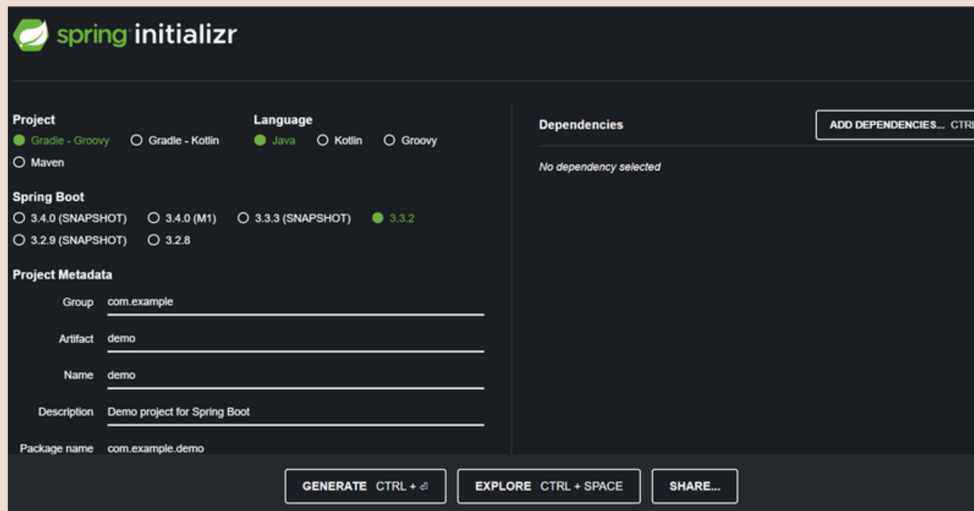


@MILAPMAGAR

Step 1: Set Up Spring Boot

using Spring Initializr.

- Website: <https://start.spring.io/>



The screenshot shows the Spring Initializr web application interface. The header features the Spring logo and the text "spring initializr". The main content area is divided into several sections:

- Project:** Includes radio buttons for "Gradle - Groovy" (selected), "Gradle - Kotlin", and "Maven".
- Language:** Includes radio buttons for "Java" (selected), "Kotlin", and "Groovy".
- Spring Boot:** Includes radio buttons for "3.4.0 (SNAPSHOT)", "3.4.0 (M1)", "3.3.3 (SNAPSHOT)", "3.3.2" (selected), and "3.2.9 (SNAPSHOT)".
- Project Metadata:** Includes input fields for "Group" (com.example), "Artifact" (demo), "Name" (demo), "Description" (Demo project for Spring Boot), and "Package name" (com.example.demo).
- Dependencies:** Includes a button "ADD DEPENDENCIES... CTRL" and the text "No dependency selected".

At the bottom, there are three buttons: "GENERATE CTRL + ⌘", "EXPLORE CTRL + SPACE", and "SHARE...".

Step 2: Create a REST Controller

```
© DemoController.java x
7  import org.springframework.web.bind.annotation.RestController;
8  public class DemoController { no usages
9      @RestController no usages
10     @RequestMapping("/api")
11     public class DemoControllers {
12
13         // GET method to fetch a message
14         @GetMapping("/hello") no usages
15         public String getHello() {
16             return "Hello, World!";
17         }
18
19         // GET method to fetch a message with a path variable
20         @GetMapping("/hello/{name}") no usages
21         public String getHelloName(@PathVariable String name) {
22             return "Hello, " + name + "!";
23         }
24
25         // POST method to create a new message
26         @PostMapping("/message") no usages
27         public String postMessage(@RequestBody String message) {
28             return "Message received: " + message;
29         }
30     }
31 }
```

Step 3: Application Entry point



```
1 package com.milap.myfirstProject;
2
3
4 import org.springframework.boot.SpringApplication;
5 import org.springframework.boot.autoconfigure.SpringBootApplication;
6
7 @SpringBootApplication
8 public class DemoApplication {
9
10     public static void main(String[] args) {
11         SpringApplication.run(DemoApplication.class, args);
12     }
13 }
14
15
```

Step 4: Run the Application

Run your Spring Boot application, and you should be able to access the endpoints:

- GET -> `http://localhost:8080/api/hello` - will return "Hello, World!"
- GET `http://localhost:8080/api/hello/{name}` will return "Hello, {name}!" (replace {name} with any string)
- POST -> `http://localhost:8080/api/message` with a JSON body will return "Message received: {your_message}"

Summary

- **REST Controllers:** Define controllers to handle HTTP requests and responses. Request Mapping: Map HTTP requests to specific handler methods using annotations.
- **GET Method:** Retrieve data from the server.
- **POST Method:** Send data to the server for creation or processing. Path Variables: Capture dynamic values from the URI. Request Body: Bind the HTTP request body to a method parameter.
- **Spring Boot Application:** Entry point to run the Spring Boot application.
- **Testing Endpoints:** Use tools like Postman or curl to test your REST APIs.

Follow for more updates



@milapeeeeyyy



@Milap-Magar