

## Creating a Rest Controller

### 👉 Setting up the Project

Step 1: Create a new Spring Boot project using Spring Initializer

The screenshot shows the Spring Initializr web interface. In the 'Project' section, 'Maven' is selected. Under 'Language', 'Java' is selected. In the 'Spring Boot' section, '3.4.4' is selected. The 'Project Metadata' section includes fields for Group (com.telusko), Artifact (spring-boot-rest), Name (spring-boot-rest), Description (Demo project for Spring Boot), and Package name (com.telusko.spring-boot-rest). The 'Dependencies' section is currently empty. At the bottom, there are 'GENERATE' and 'EXPLORE' buttons.

Step 2: Add the necessary dependencies:

The screenshot shows the Spring Initializr web interface with the 'Spring Web' dependency added. The 'Dependencies' section now lists 'Spring Web [WEB]' and 'Lombok [DEVELOPER TOOLS]'. The 'Spring Web' dependency is described as 'Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.' The 'Lombok' dependency is described as 'Java annotation library which helps to reduce boilerplate code.' All other project settings remain the same as in the previous screenshot.

**Step 3:** Download, unzip, and open the project in your IDE (IntelliJ, Eclipse, or STS)

**Step 4:** Import the model, repository, and service classes from your previous codebase

## 👉 Creating the REST Controller

Now, let's create a controller that will handle HTTP requests:

- Create a new class called `JobRestController` in the controller package
- Annotate it with `@RestController`
- Inject the `JobService` using `@Autowired`
- Create a method called `getAllJobs()` that returns a list of job posts
- Annotate the method with `@GetMapping("jobPosts")`

**Example:**

```
● ● ●

package com.telusko.springbootrest.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import com.telusko.springbootrest.model.JobPost;
import com.telusko.springbootrest.service.JobService;

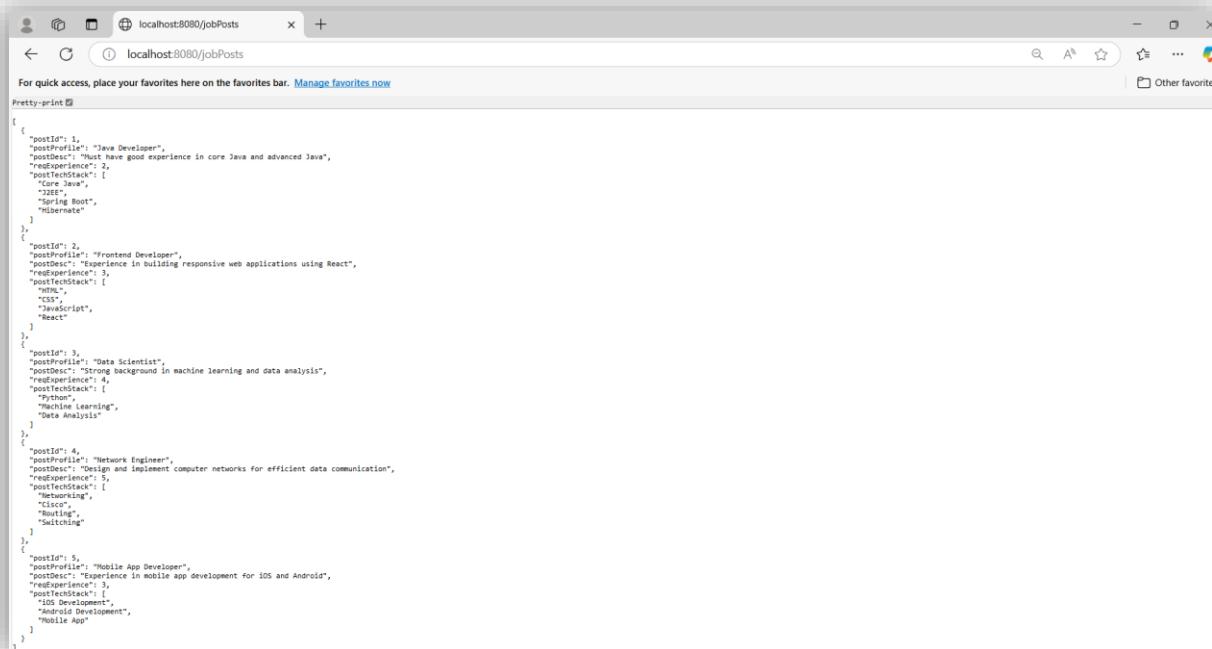
//Controller class to handle REST API requests related to job posts.
@RestController
public class JobRestController {

    // Injecting JobService to access business logic related to job posts
    @Autowired
    private JobService service;

    /**
     * Handles GET requests for retrieving all job posts.
     *
     * @return List of JobPost objects
     */
    @GetMapping("jobPosts")
    public List<JobPost> getAllJobs() {
        // Fetching all job posts using the service layer
        return service.getAllJobs();
    }
}
```

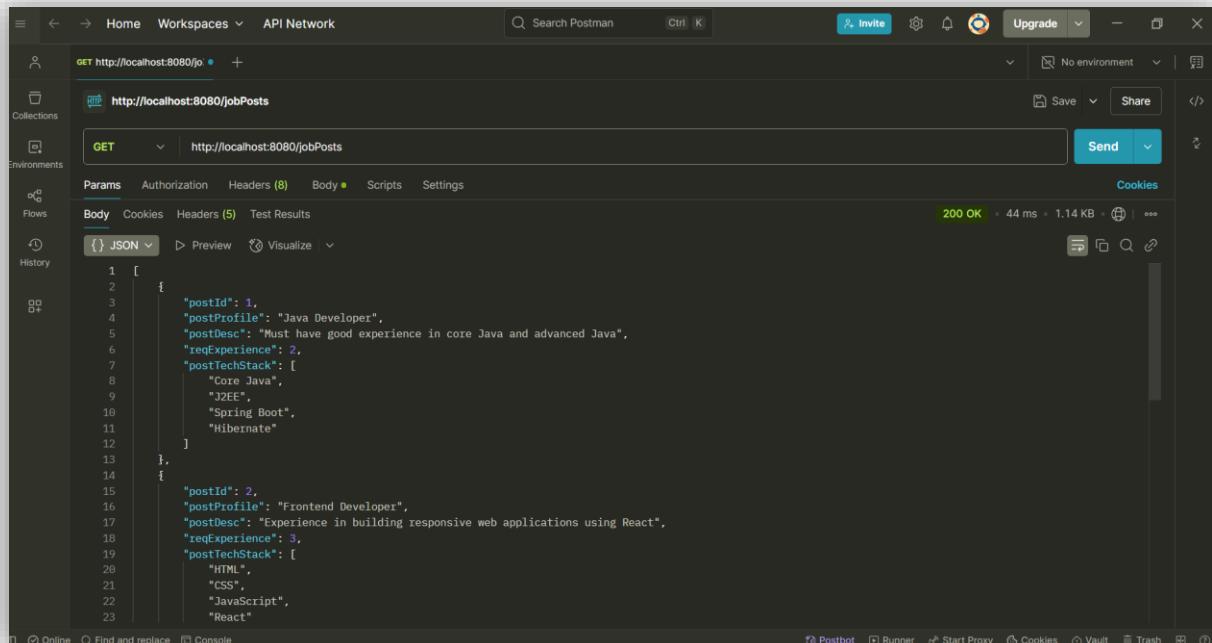
➤ After running your Spring Boot application:

1. Open your browser and go to: <http://localhost:8080/jobPosts>



```
[{"postId": 1, "postProfile": "Java Developer", "postDesc": "Must have good experience in core Java and advanced Java", "reqExperience": 2, "postTechSkills": ["Core Java", "J2EE", "Spring Boot", "Hibernate"], "postTechStack": []}, {"postId": 2, "postProfile": "Frontend Developer", "postDesc": "Experience in building responsive web applications using React", "reqExperience": 1, "postTechSkills": ["HTML", "CSS", "JavaScript", "React"], "postTechStack": []}, {"postId": 3, "postProfile": "Data Scientist", "postDesc": "Strong background in machine learning and data analysis", "reqExperience": 4, "postTechSkills": ["Python", "Machine Learning", "Data Analysis"], "postTechStack": []}, {"postId": 4, "postProfile": "Network Engineer", "postDesc": "Design and implement computer networks for efficient data communication", "reqExperience": 5, "postTechSkills": ["Networking", "Cisco", "Routing", "Switching"], "postTechStack": []}, {"postId": 5, "postProfile": "Mobile App Developer", "postDesc": "Experience in mobile app development for iOS and Android", "reqExperience": 3, "postTechSkills": ["iOS Development", "Android Development", "React Native"], "postTechStack": []}]
```

2. Or use Postman with the same URL to test the GET request



GET http://localhost:8080/jobPosts

200 OK

```
[{"postId": 1, "postProfile": "Java Developer", "postDesc": "Must have good experience in core Java and advanced Java", "reqExperience": 2, "postTechSkills": ["Core Java", "J2EE", "Spring Boot", "Hibernate"], "postTechStack": []}, {"postId": 2, "postProfile": "Frontend Developer", "postDesc": "Experience in building responsive web applications using React", "reqExperience": 1, "postTechSkills": ["HTML", "CSS", "JavaScript", "React"], "postTechStack": []}, {"postId": 3, "postProfile": "Data Scientist", "postDesc": "Strong background in machine learning and data analysis", "reqExperience": 4, "postTechSkills": ["Python", "Machine Learning", "Data Analysis"], "postTechStack": []}, {"postId": 4, "postProfile": "Network Engineer", "postDesc": "Design and implement computer networks for efficient data communication", "reqExperience": 5, "postTechSkills": ["Networking", "Cisco", "Routing", "Switching"], "postTechStack": []}, {"postId": 5, "postProfile": "Mobile App Developer", "postDesc": "Experience in mobile app development for iOS and Android", "reqExperience": 3, "postTechSkills": ["iOS Development", "Android Development", "React Native"], "postTechStack": []}]
```

## 👉 Understanding `@RestController`

`@RestController` is a special annotation in Spring that combines two other annotations:

- `@Controller` - Marks the class as a web controller, capable of handling HTTP requests
- `@ResponseBody` - Tells Spring to convert the return value directly to JSON/XML response

The key difference between a regular `@Controller` and a `@RestController` is:

- Regular controllers typically return view names (like JSP pages)
- REST controllers return data objects directly, which Spring automatically converts to JSON

This makes `@RestController` perfect for building APIs that send and receive data rather than rendering HTML pages. When your frontend React application makes a request to `/jobPosts`, it will receive the job data in JSON format, which it can then display to the user.