Palash Bajpai

Spring Boot

Interview Questions

Table of Contents

[Q1. What is RestController and how it is different from controller? 2](#_Toc197068520)

[Q2. How to show html page via controller? 3](#_Toc197068521)

## Q1. What is RestController and how it is different from controller?

@RestController is a **convenience annotation** that is equal to:

@Controller + @ResponseBody

It tells Spring:

* This class is a controller.
* All methods inside will return data **directly in the response body** (usually as JSON) instead of views.

**@Controller**

* @Controller is used to define a Spring MVC Controller class.
* It's a specialization of @Component, so Spring will detect it during component scanning and manage it as a Spring Bean.
* Used to handle web requests.
* Typically returns views (JSP, Thymeleaf, etc.) using a ModelAndView or by returning a String (view name).

**@ResponseBody**

@ResponseBody is a Spring annotation used to bind the return value of a method directly to the HTTP response body, rather than rendering a view (like a JSP or HTML page).

Without @ResponseBody: If you return a String in a controller method, Spring assumes it's the name of a view (like home.jsp) to be rendered.

With @ResponseBody: Spring bypasses the view resolver and sends the return value directly in the HTTP response body, usually as JSON (if using Jackson) or XML (if using JAXB), depending on the configured message converters.

With RestController

@RestController  
public class HelloController {  
 @GetMapping("/hello")  
 public String hello(){  
 return " Hello World";  
 }  
}

Equivalent code with just Controller is

@Controller  
public class HelloController {  
 @GetMapping("/hello")  
 @ResponseBody  
 public String hello(){  
 return " Hello World";  
 }  
}

## Q2. How to show html page via controller?

If you're using @Controller (not @RestController) and want to return HTML pages, then you’ll be using Thymeleaf or another templating engine like JSP (though JSP is discouraged with Spring Boot). The default behavior for Spring Boot with Thymeleaf is as follows:

1. Add Thymeleaf dependency

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-thymeleaf</artifactId>  
</dependency>

2. Put your hellohello.html files inside this folder: src/main/resources/templates/

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head>  
 <title>Hello Page</title>  
</head>  
<body>  
<h1>Hello, World!</h1>  
</body>  
</html>

3. Create Controller

@Controller  
public class HelloController {  
 @GetMapping("/hello")  
 public String hello(){  
 return "hello"; // Maps to hello.html in templates folder  
 }  
}

4. To pass data to html page via controller, we can use Model object.

import org.springframework.ui.Model;  
  
@Controller  
public class HelloController {  
 @GetMapping("/hello")  
 public String hello(Model model){  
 model.addAttribute("user","Palash");  
 return "hello"; // Maps to hello.html in templates folder  
 }  
}

in html:

</head>  
<body>  
 <h1 th:text="'Hello, ' + ${user} + '!'">Hello, Guest!</h1>  
</body>  
</html>

* th:text="'Hello, ' + ${username} + '!'"

Renders: Hello, Palash!

* If the model value is missing, the fallback content (e.g. Hello, Guest!) is displayed.