Difference between @Controller and @RestController in Spring MVC?

@Controller is used to declare common web controllers which can return HTTP response but @RestController is used to create controllers for REST APIs which can return JSON.

JAVINPAUL AND SOMA

DEC 14, 2023 · PAID

Hello friends, if you are preparing for a Java developer interview with Spring experience then you may have come across the *"difference between @Controller and @RestController annotation"* in Spring Boot and Spring Framework in general. It's one of the popular spring questions and I have seen it many times.

In Spring MVC, both `@Controller` and `@RestController` annotations are used to define web controllers as per the MVC Design pattern. A controller is responsible for handling HTTP requests and returning HTTP responses to the client.

> The main difference between these two annotation is how they handle client's request and when they are used

For example, *@Controller* annotation is there from day 1 and it is used to mark a class as a web controller to process HTTP requests and return a view name, which is then resolved by a view resolver to generate the final HTML view.

On the other hand `@RestController` annotation was added in later Spring versions like Spring 3.4 to increase support for REST API development. In the case of REST API, instead of returning HTML, you will probably like to return JSON or XML as your client is no more human but a machine.

If you want to return a JSON or XML from a Spring MVC controller then you need to add @ReseponseBody annotation to each of the Controller methods and it seems overkill while implementing REST APIs using Spring and Spring Boot.

Designers and developers of Spring Framework recognized this shortcoming and then added a new annotation called **@RestController** in the Spring 3.4 version.

The `@RestController` annotation is a combination of the `@Controller` and `@ResponseBody` annotations and you can use it to implement REST APIs in Java and Spring Boot.

> The key difference between `@Controler` and `@RestController` annotation is @ResponseBody annotation, `@Controler` does not automatically add the `@ResponseBody` annotation to all of the controller's methods, which means that you need to add it to each method individually if you want to return a JSON or XML response.`@RestController` automatically adds the `@ResponseBody` annotation to all of the controller's methods.

Now let's see examples of both @Controller and @RestController annotation in Spring framework

@Controller and @RestController Example in Spring Boot

Here is a simple example of `@Controller` annotation which is used to return a response for simple HTTP requests on "`/hello`" path:

```
@Controller
public class MyController {

    @Autowired
    private MyService myService;

    @RequestMapping("/hello")
    public String sayHello(Model model) {
        model.addAttribute("message", myService.getHelloMessage());
        return "hello";
    }
}
```

In this example, the `MyController` class is annotated with `@Controller` annotation and has a single method, `sayHello()`, which is mapped to the `/hello` URL.

The method takes a `Model` object as a parameter and adds an attribute called `"message"` to it. The method then returns a string "hello", which is the name of the view to be rendered.

Now, let's see an example of `@RestController` annotation from Spring Framework, which I used to respond to REST API

```
@RestController
public class MyRestController {

    @Autowired
    private MyService myService;

    @RequestMapping("/greeting")
    public Greeting getGreeting() {
        return myService.getGreeting();
    }
}
```

In this example, the `MyRestController` class is annotated with `@RestController` and has a single method, `getGreeting()`, which is mapped to the `/greeting` URL.

The method returns an object of the `Greeting` class, which will be **automatically converted to a JSON or XML representation** and sent as a response to the client.

It is also worth noting that the `Greeting` class should have `getters` and `setters` for the data that you want to be included in the response, otherwise, it will not be serialized properly or converted to JSON.

Things to keep in mind while using @Controller and @RestController annotations in Spring Boot

Here are a few things to keep in mind while using the `@Controller` and `@RestController` annotations:

- When using the @Controller annotation, you will need to use the @ResponseBody annotation to return JSON or XML data, otherwise, the response will be treated as a view name, and a view resolver will try to resolve it while the @ResponseBody annotation is not necessary as it is already included in the annotation.

- When using the @RestController annotation, you can also use the *@ResponseStatus annotation to set the HTTP status code of the response* and the @RequestBody annotation to bind the request body to a method parameter.
- And while using the @RestController annotation, you can use the @RequestMapping annotation to set the URL path to map the methods.
- Another thing worth noting is that, If you are returning a complex object that contains child objects, make sure all the child object classes have getters and setters, otherwise, they will not be serialized properly.

That's all about the *difference between @Controller and @RestController annotation in Spring Boot.* Just remember that @Controller is used to create web controllers that return views, which is further resolved by view resolver, while `@RestController` is used to create web services that return JSON or XML data. It's also worth noting.

And, if you are preparing for Java interviews you can also check my **Java + Spring Interview + SQL Bundle on GUmroad**, use code **friends20** to get a 20% discount also

5 Likes · 1 Restack

Previous

Next

A guest post by
Soma
Java and React Developer

Subscribe to Soma

Comments

Write a comment...