Difference between @GetMapping and @RequestMapping Spring?

JAVINPAUL AND SOMA

APR 12, 2024 • PAID

Hello friends , if you are preparing for Java Developer and Spring Developer interviews then you must prepare for questions like difference between X and Y like **@bean vs @component** or @ReqestMapping or @GetMapping, those are very popular on Java interviews .

From last few episodes I have been sharing Java questions so one of my paid subscriber also ask if I can also share Spring related questions more, which I thought a good idea, so here we are.

In the past, I have also shared few questions on Spring boot like **difference between RestController and Controller** and in this article, we will take a look at another set of popular Spring annotations which is commonly related to processing HTTP request.

In Spring Framework, you will find variety of **annotations** that play a crucial role in shaping the behavior of their applications.

Two such annotations that are frequently used when dealing with web controllers are `@RequestMapping` and `@GetMapping`.

These annotations provide a way to define the mapping of incoming HTTP requests to methods within the controller, allowing for seamless handling of client requests.

However, while they might appear to be similar at first glance, they serve distinct purposes and offer specific functionalities for example @GetMapping is only for HTTP GET request while @RequestMapping can be used to process any HTTP methods like POST, PUT, GET and even DELETE. In this article, we'll delve into the depths of these annotations, uncovering their differences, use cases, and benefits.

Now, that you know the basic difference, its time for deep dive.

Introduction to `@RequestMapping` and `@GetMapping` in Spring MVC Web Framework

Before delving into the differences, let's start by understanding what each annotation stands for and what purpose it serves.

1. @RequestMapping

`@RequestMapping` is a versatile annotation in the Spring Framework that maps HTTP requests to methods in your controller class. It provides a mechanism to handle various HTTP methods such as GET, POST, PUT, DELETE, etc., and it allows you to define the URI patterns that trigger the execution of the associated methods.

Here's a basic example of how `@RequestMapping` can be used:

```
@Controller
@RequestMapping("/myapp")
public class MyController {

    @RequestMapping("/hello")
    public String sayHello() {
        return "Hello, world!";
    }
}
```

In this example, any request made to the URI `/myapp/hello` will trigger the `sayHello` method, which returns the string "Hello, world!".

2. @GetMapping

`@GetMapping` is a specialized form of the `@RequestMapping` annotation that is focused on handling GET requests exclusively. It was introduced in Spring 4.3 as a shorthand way of specifying that a method should handle GET requests without the need to explicitly specify the HTTP method.

Here's how `@GetMapping` can be used:

```
@Controller
@RequestMapping("/myapp")
public class MyController {

    @GetMapping("/hello")
    public String sayHello() {
        return "Hello, world!";
    }
}
```

In this example, the `sayHello` method is explicitly marked as a handler for GET requests using the `@GetMapping` annotation.

Difference between @GetMapping and @RequestMapping in Spring MVC

Now that we have a basic understanding of both annotations, let's explore the key differences between `@RequestMapping` and `@GetMapping`.

1. Purpose and Specialization

The primary difference between these two annotations lies in their purpose and specialization. While `@RequestMapping` is a general-purpose annotation that can handle requests of any HTTP method, `@GetMapping` is tailored specifically for GET requests.

This specialization enhances the clarity and readability of the code, making it easier to understand the intended purpose of the annotated method.

2. Method Signatures

Another notable distinction is the method signatures used with these annotations. With `@RequestMapping`, you need to explicitly specify the HTTP method using the `method` attribute. For example:

```
@RequestMapping(value = "/hello", method = RequestMethod.GET)
public String sayHello() {
    return "Hello, world!";
}
```

On the other hand, `@GetMapping` omits the need for the `method` attribute altogether, as it's implicitly tied to GET requests:

```
@GetMapping("/hello")
public String sayHello() {
    return "Hello, world!";
}
```

The reduction in the number of attributes can lead to cleaner and more concise code, enhancing code readability.

3. Intuitive Mapping

One of the primary benefits of using `@GetMapping` is the intuitiveness it brings to your codebase. When you encounter a method annotated with `@GetMapping`, it's immediately clear that this method is designed to handle GET requests.

This explicitness improves the code's maintainability and reduces the chances of errors caused by mismatched HTTP methods.

4. Avoiding Ambiguity

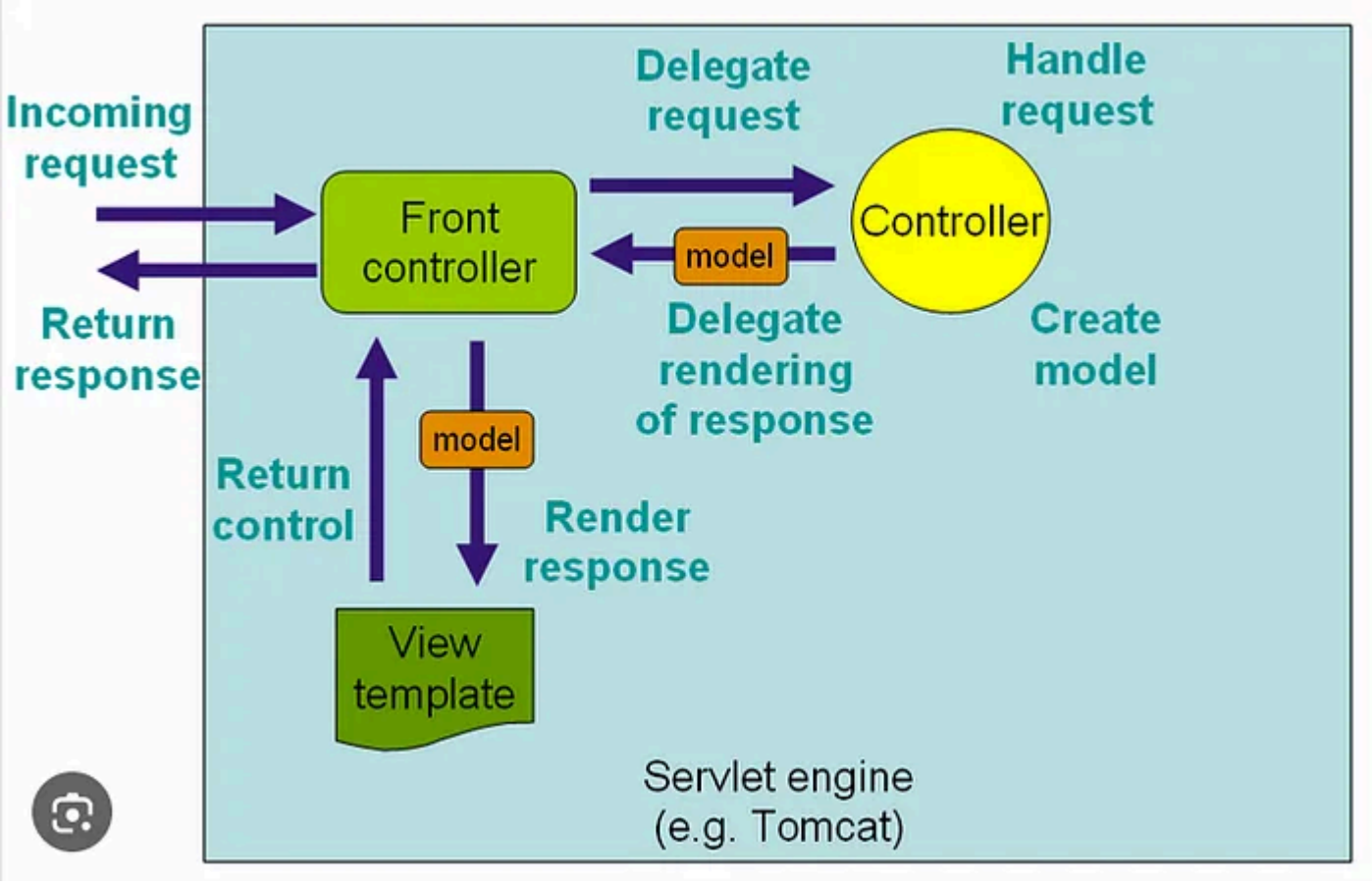
When dealing with complex controllers that handle multiple HTTP methods, using `@GetMapping` can help prevent ambiguity in method mappings.

By explicitly marking a method as a handler for GET requests, you avoid unintentional overlaps with methods that handle other HTTP methods.

This can be particularly useful in large codebases with multiple developers working collaboratively.

5. Readability and Documentation

In terms of code readability and documentation, `@GetMapping` provides self-contained and concise information about the method's purpose. Developers can easily identify the supported HTTP method without the need to inspect additional attributes.



How to Choose Between `@RequestMapping` and `@GetMapping` Annotations?

The decision to use either `@RequestMapping` or `@GetMapping` depends on the context of your application and your coding preferences. Here are some considerations to help you make an informed choice:

Use `@RequestMapping` When:

- You need to handle multiple HTTP methods within the same method.
- Your codebase requires maximum flexibility in terms of method selection.
- You are working with older versions of Spring that do not support `@GetMapping`.

Use `@GetMapping` When:

- Your method is intended to handle GET requests exclusively.
- Code readability and intent clarity are important to you.
- You want to take advantage of the shorter method signature and reduced attribute usage.
- You are using Spring 4.3 or newer, where `@GetMapping` is available.

Conclusion

That's all about the difference between `@RequestMapping` and @GetMapping annotation in Spring. In the Spring Framework, `@RequestMapping` and `@GetMapping` serve as crucial annotations for mapping incoming HTTP requests to controller methods.

The key difference between the two is that **@RequestMapping is used for all kind of HTTP methods while `@GetMapping` offers a specialized, succinct, and intuitive approach for handling GET requests.**

While they share the core functionality of routing requests, they cater to different scenarios and developer preferences. `@RequestMapping` is a versatile choice suitable for a wide range of HTTP methods and mappings, whereas `@GetMapping` offers a specialized, succinct, and intuitive approach for handling GET requests.

By understanding the differences between these annotations, developers can make informed decisions when designing their controller methods, ensuring that their codebase remains well-organized, maintainable, and comprehensible.

The choice between `@RequestMapping` and `@GetMapping` ultimately comes down to the specific requirements of your application and your desire for code clarity and efficiency. With

this knowledge in hand, you're well-equipped to harness the power of Spring annotations to build robust and effective web applications.

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

All the best with your Java interviews !!.

3 Likes · 1 Restack

← Previous

Next →

A guest post by

Soma

Java and React Developer

Subscribe to Soma

1 Comment

Write a comment...

Soma Apr 12 Author

Thank you for publishing

LIKE REPLY SHARE