

( / )

# Spring @RequestParam Annotation

Last updated: January 9, 2022



Written by: baeldung (<https://www.baeldung.com/author/baeldung>)

**Spring MVC** (<https://www.baeldung.com/category/spring/spring-web/spring-mvc>)

**Spring Annotations** (<https://www.baeldung.com/tag/spring-annotations>)

**Spring Core Basics** (<https://www.baeldung.com/tag/spring-core-basics>)

**Spring MVC Basics** (<https://www.baeldung.com/tag/spring-mvc-basics>)

( / )

**Get started with Spring and Spring Boot, through the *Learn Spring* course:**

**>> CHECK OUT THE COURSE ([/ls-course-start](#))**

## 1. Overview

In this quick tutorial, we'll explore Spring's *@RequestParam* annotation and its attributes.

**Simply put, we can use *@RequestParam* to extract query parameters, form parameters, and even files from the request.**

### Further reading:

#### **Spring @RequestMapping New Shortcut Annotations ([/spring-new-requestmapping-shortcuts](#))**

In this article, we introduce different types of @RequestMapping shortcuts for quick web development using traditional Spring MVC framework.

**Read more ([/spring-new-requestmapping-shortcuts](#)) →**

#### **The Spring @Controller and @RestController Annotations ([/spring-controller-vs-restcontroller](#))**

Learn about the differences between @Controller and @RestController annotations in Spring MVC.

**Read more ([/spring-controller-vs-restcontroller](#)) →**

## 2. A Simple Mapping

Let's say that we have an endpoint `/api/foos` that takes a query parameter called `id`:

```
@GetMapping("/api/foos")
@ResponseBody
public String getFoos(@RequestParam String id) {
    return "ID: " + id;
}
```

In this example, we used `@RequestParam` to extract the `id` query parameter. A simple GET request would invoke `getFoos`:

```
http://localhost:8080/spring-mvc-basics/api/foos?id=abc
----
ID: abc
```

Next, **let's have a look at the annotation's attributes: *name*, *value*, *required*, and *defaultValue*.**

## 3. Specifying the Request Parameter Name

In the previous example, both the variable name and the parameter name are the same.

**Sometimes we want these to be different, though.** Or, if we aren't using Spring Boot, we may need to do special compile-time configuration or the parameter names won't actually be in the bytecode.

Fortunately, **we can configure the `@RequestParam` name using the *name* attribute:**

```
@PostMapping("/api/foos")
@ResponseBody
public String addFoo(@RequestParam(name = "id") String fooId,
    @RequestParam String name) {
    return "ID: " + fooId + " Name: " + name;
}
```

We can also do `@RequestParam(value = "id")` or just `@RequestParam("id")`.

## 4. Optional Request Parameters

Method parameters annotated with `@RequestParam` are required by default. This means that if the parameter isn't present in the request, we'll get an error:

```
GET /api/foos HTTP/1.1
-----
400 Bad Request
Required String parameter 'id' is not present
```

**We can configure our `@RequestParam` to be optional, though, with the `required` attribute:**

```
@GetMapping("/api/foos")
@ResponseBody
public String getFoos(@RequestParam(required = false) String id) {
    return "ID: " + id;
}
```

In this case, both:

```
http://localhost:8080/spring-mvc-basics/api/foos?id=abc
-----
ID: abc
```

and

```
http://localhost:8080/spring-mvc-basics/api/foos  
----  
ID: null
```

will correctly invoke the method.

**When the parameter isn't specified, the method parameter is bound to *null*.**

## 4.1. Using Java 8 *Optional*

Alternatively, we can wrap the parameter in *Optional* (*/java-optional*).

```
@GetMapping("/api/foos")  
@ResponseBody  
public String getFoos(@RequestParam Optional<String> id){  
    return "ID: " + id.orElseGet(() -> "not provided");  
}
```

In this case, **we don't need to specify the *required* attribute.**

And the default value will be used if the request parameter is not provided:

```
http://localhost:8080/spring-mvc-basics/api/foos  
----  
ID: not provided
```

## 5. A Default Value for the Request Parameter

We can also set a default value to the *@RequestParam* by using the *defaultValue* attribute:

```
@GetMapping("/api/foos/{id}")  
@ResponseBody  
public String getFoos(@RequestParam(defaultValue = "test") String id)  
{  
    return "ID: " + id;  
}
```

This is like *required=false*, in that the user no longer needs to supply the parameter:

```
http://localhost:8080/spring-mvc-basics/api/foos  
----  
ID: test
```

Although, we are still okay to provide it:

```
http://localhost:8080/spring-mvc-basics/api/foos?id=abc  
----  
ID: abc
```

Note that when we set the *defaultValue* attribute, *required* is indeed set to *false*.

## 6. Mapping All Parameters

We can also have multiple parameters without defining their names or count by just using a *Map*:

```
@PostMapping("/api/foos")  
@ResponseBody  
public String updateFoos(@RequestParam Map<String,String> allParams) {  
    return "Parameters are " + allParams.entrySet();  
}
```

which will then reflect back any parameters sent:

```
curl -X POST -H 'name=abc' -F '{id=123}' http://localhost:8080/spring-  
mvc-basics/api/foos  
-----  
Parameters are {[name=abc], [id=123]}
```

## 7. Mapping a Multi-Value Parameter

A single *@RequestParam* can have multiple values:

```
@GetMapping("/api/foos")  
@ResponseBody  
public String getFoos(@RequestParam List<String> id) {  
    return "IDs are " + id;  
}
```

And Spring MVC will map a comma-delimited *id* parameter:

```
http://localhost:8080/spring-mvc-basics/api/foos?id=1,2,3  
-----  
IDs are [1,2,3]
```

or a list of separate *id* parameters:

```
http://localhost:8080/spring-mvc-basics/api/foos?id=1&id=2  
-----  
IDs are [1,2]
```

## 8. Conclusion

In this article, we learned how to use *@RequestParam*.

The full source code for the examples can be found in the GitHub project (<https://github.com/eugenp/tutorials/tree/master/spring-web-modules/spring-mvc-basics-5>).

[\(/\)](#)

**Get started with Spring and Spring Boot, through the *Learn Spring* course:**

**>> CHECK OUT THE COURSE ([/ls-course-end](#))**



Learning to build your API  
**with Spring?**

**Download the E-book ([/rest-api-spring-guide](#))**

---

Comments are closed on this article!

## COURSES

[ALL COURSES \(/ALL-COURSES\)](#)



ALL BULK COURSES (/ALL-BULK-COURSES)

ALL BULK TEAM COURSES (/ALL-BULK-TEAM-COURSES)

THE COURSES PLATFORM (HTTPS://COURSES.BAELDUNG.COM)

## SERIES

JAVA "BACK TO BASICS" TUTORIAL (/JAVA-TUTORIAL)

JACKSON JSON TUTORIAL (/JACKSON)

APACHE HTTPCLIENT TUTORIAL (/HTTPCLIENT-GUIDE)

REST WITH SPRING TUTORIAL (/REST-WITH-SPRING-SERIES)

SPRING PERSISTENCE TUTORIAL (/PERSISTENCE-WITH-SPRING-SERIES)

SECURITY WITH SPRING (/SECURITY-SPRING)

SPRING REACTIVE TUTORIALS (/SPRING-REACTIVE-GUIDE)

## ABOUT

ABOUT BAELDUNG (/ABOUT)

THE FULL ARCHIVE (/FULL\_ARCHIVE)

EDITORS (/EDITORS)

JOBS (/TAG/ACTIVE-JOB/)

OUR PARTNERS (/PARTNERS)

PARTNER WITH BAELDUNG (/ADVERTISE)

TERMS OF SERVICE (/TERMS-OF-SERVICE)

PRIVACY POLICY (/PRIVACY-POLICY)

COMPANY INFO (/BAELDUNG-COMPANY-INFO)

CONTACT (/CONTACT)