

With Spring 3.0, can I make an optional path variable?

With Spring 3.0, can I have an optional path variable?

For example

```
@RequestMapping(value = "/json/{type}", method = RequestMethod.GET)
public @ResponseBody TestBean testAjax(
    HttpServletRequest req,
    @PathVariable String type,
    @RequestParam("track") String track) {
    return new TestBean();
}
```

Here I would like `/json/abc` or `/json` to call the same method.
One obvious workaround declare `type` as a request parameter:

```
@RequestMapping(value = "/json", method = RequestMethod.GET)
public @ResponseBody TestBean testAjax(
    HttpServletRequest req,
    @RequestParam(value = "type", required = false) String type,
    @RequestParam("track") String track) {
    return new TestBean();
}
```

and then `/json?type=abc&track=aa` or `/json?track=rr` will work

[spring](#) [rest](#)

edited Aug 31 '16 at 14:58



[lambda](#)

1,960 1 15 24

asked Feb 4 '11 at 23:58



[Shamik](#)

3,029 9 43 66

6 Answers

You can't have optional path variables, but you can have two controller methods which call the same service code:

```
@RequestMapping(value = "/json/{type}", method = RequestMethod.GET)
public @ResponseBody TestBean typedTestBean(
    HttpServletRequest req,
    @PathVariable String type,
    @RequestParam("track") String track) {
    return getTestBean(type);
}

@RequestMapping(value = "/json", method = RequestMethod.GET)
public @ResponseBody TestBean testBean(
    HttpServletRequest req,
    @RequestParam("track") String track) {
    return getTestBean();
}
```

edited Aug 31 '16 at 14:59



[lambda](#)

1,960 1 15 24

answered Feb 5 '11 at 0:06



[earldouglas](#)

9,964 4 28 46

This is surely a nice way of doing it. – [Shamik](#) Feb 5 '11 at 1:48

3 @Shamik: This is a compelling reason *not* to use path variables, in my opinion. The combinatorial proliferation can quickly get out of hand. – [skaffman](#) Feb 5 '11 at 11:40

7 Actually not because the path can't be that complex while being filled up with optional components. If you have more than one or max two optional path elements you should seriously consider switching a few of them to request parameters. – [Patrick Cornelissen](#) Apr 23 '12 at 10:44

And for some people, having the second controller method call the first controller method may work as well, if for instance the differing parameter can be provided by some other means – [chrismarx](#) May 2 '12 at 14:06

1 Please consider updating your answer, instead of creating two controller methods in newer version of Spring we may just use `@RequestMapping` with two values as in: stackoverflow.com/questions/17821731/... – [csharpfolk](#) Nov 6 '16 at 18:12

It's not well known that you can also inject a Map of the path variables using the

@PathVariable annotation. I'm not sure if this feature is available in Spring 3.0 or if it was added later, but here is another way to solve the example:

```
@RequestMapping(value={ "/json/{type}", "/json" }, method=RequestMethod.GET)
public @ResponseBody TestBean typedTestBean(
    @PathVariable Map<String, String> pathVariables,
    @RequestParam("track") String track) {

    if (pathVariables.containsKey("type")) {
        return new TestBean(pathVariables.get("type"));
    } else {
        return new TestBean();
    }
}
```

answered Mar 13 '15 at 17:08



Paul Wardrip
744 4 3

1 thanks, In spring 4.2 works as expected – Edgar Jun 25 '15 at 12:25

I use it all the time .This comes handy when i want a single method to handle different uri types ex:{
"/json/{type}", "/json/{type}/{xyz}", "/json/{type}/{abc}", "/json/{type}/{abc}/{something}", "/json" } – Vaibs Aug 23 '17 at 13:21

If you are using Spring 4.1 and Java 8 you can use `java.util.Optional` which is supported in `@RequestParam` , `@PathVariable` , `@RequestHeader` and `@MatrixVariable` in Spring MVC -

```
@RequestMapping(value = {"/json/{type}", "/json" }, method = RequestMethod.GET)
public @ResponseBody TestBean typedTestBean(
    @PathVariable Optional<String> type,
    @RequestParam("track") String track) {
    if (type.isPresent()) {
        //type.get() will return type value
        //corresponds to path "/json/{type}"
    } else {
        //corresponds to path "/json"
    }
}
```

answered Jan 8 '16 at 6:52



Aniket Thakur
35.7k 21 161 182

you sure this would work? Your own answer [here](#) suggests that the controller won't be hit if {type} is absent from the path. I think PathVariable Map would be a better approach, or using separate controllers. – Anshul Tiwari Oct 17 '16 at 13:39

3 Yes if you just have `"/json/{type}"` and `type` is not present it will not be hit (as my linked answer suggests) but here you have `value = {"/json/{type}", "/json" }` . So if any one matches controller method will be hit. – Aniket Thakur Oct 17 '16 at 19:04

Is it possible that the same value, etc., `type` would be `RequestParam` and `RequestParam` too? – zyгимantus Feb 11 '17 at 22:15

It works, but anyway the controller's function won't be called because it expects a parameter there – EliuX Mar 30 '17 at 17:15

4 This works, and since Spring 4.3.3, you can also go with `@PathVariable(required = false)` and get null if the variable is not present. – Nicolai Ehemann Oct 5 '17 at 14:21

You could use a :

```
@RequestParam(value="somvalue",required=false)
```

for optional params rather than a `PathVariable`

answered Aug 20 '12 at 20:40



Maleck13
1,114 9 15

1 This is version specific, it seems. No go for Spring 3. – Stu Thompson Feb 6 '13 at 18:08


4 Currently using this method for a spring 3.1 project, and the docs say that it works for 2.5+, so it definitely works for Spring 3. EDIT: [source](#). – Evan Byrne Mar 22 '13 at 16:21

18 True, but this is not what the question is about. Using *request* parameters is indeed mentioned in the question as "*One obvious workaround*", but the question itself is about *path* parameters. This is not a solution for optional path parameters. – Arjan Apr 14 '13 at 10:29

8 PathVariable and RequestParam are different. – Timeless Mar 16 '15 at 1:43

Check this [Spring 3 WebMVC - Optional Path Variables](#). It shows an article of making an extension to AntPathMatcher to enable optional path variables and might be of help. All credits to [Sebastian Herold](#) for posting the article.

answered Oct 10 '14 at 17:42

 [Uresh Kuruhuri](#)

882 7 14

```
$.ajax({
    type : 'GET',
    url : '${pageContext.request.contextPath}/order/lastOrder',
    data : {partyId : partyId, orderId :orderId},
    success : function(data, textStatus, jqXHR) {});

@RequestMapping(value = "/lastOrder", method=RequestMethod.GET)
public @ResponseBody OrderBean lastOrderDetail(@RequestParam(value="partyId") Long
partyId,@RequestParam(value="orderId",required=false) Long orderId,Model m ) {}
```

edited Oct 13 '15 at 6:44

 [Soner Gönül](#)

75.9k 24 134 249

answered Oct 13 '15 at 6:36

 [Ankush Mundada](#)

1

3 You might want to edit in some text in your answer, explaining *why* you think this contributes to solving the issue at hand (4 years later). – [Qirel](#) Oct 13 '15 at 6:54

Wouldn't help. The answer is completely wrong. – [Phil](#) Sep 7 '17 at 0:57

protected by [Aniket Thakur](#) Jan 8 '16 at 6:20

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 [reputation](#) on this site (the [association bonus](#) does not count).

Would you like to answer one of these [unanswered questions](#) instead?