# Request Method Not Supported (405) in Spring

Last modified: December 23, 2020

by baeldung

REST Spring MVC

Spring MVC Basics



## 1. Overview

This quick article is focused on a common error – 'Request Method not Supported – 405' – that developers face while exposing their APIs for specific HTTP verbs, with Spring MVC.

Naturally, we'll also discuss the common causes of this error.

## 2. Request Method Basics

Before moving towards the common problem, if you're just starting to learn about Spring MVC, here's a good intro article to start with.

Let's also have a very quick look at the basics – and understand the request methods supported by Spring and some of the common classes of interest here.

In a highly simplified way, MVC HTTP methods are basic operations that a request can trigger on the server. For example, some methods *fetch* the data from the server, some *submit* data to the server, some might *delete* the data etc.

The *@RequestMapping annotation* specifies the supported methods for the request.

Spring declares all the supported request methods under an enum *RequestMethod*; it specifies the standard *GET, HEAD, POST, PUT, PATCH, DELETE, OPTIONS, TRACE* verbs.

The Spring *DispatcherServlet* supports all of them by default except *OPTIONS* and *TRACE*; *@RequestMapping* uses the *RequestMethod enum* to specifies which methods are supported.

# 3. Simple MVC Scenario

Now, let's have a look at a code example that maps all HTTP methods:

```
@RestController
@RequestMapping(value="/api")
public class RequestMethodController {

    @Autowired
    private EmployeeService service;

    @RequestMapping(value = "/employees", produces = "application/json")
    public List<Employee> findEmployees()
      throws InvalidRequestException {
        return service.getEmployeeList();
    }
}
```

Notice how the example declares *findEmployee()* method. It doesn't specify any specific request method, which means this URL supports all default methods.

We can request the API using different supported methods, for example, using curl:

```
$ curl --request POST http://localhost:8080/api/employees
[{"id":100,"name":"Steve Martin","contactNumber":"333-777-999"},
{"id":200,"name":"Adam Schawn","contactNumber":"444-111-777"}]
```

Naturally, we can send the request in multiple ways – via a simple *curl* command, Postman, AJAX, etc.

And, of course, we're expecting to get the *200 OK* response, if the request is correctly mapped and successful.

# 4. Problem Scenario – the HTTP 405

But, what we're discussing here is – of course – the scenarios when the request won't be successful.

'*405 Method Not Allowed* is one of the most common errors we observe while working with Spring requests.

Let's have a look at what happens if we specifically define and handle GET requests in Spring MVC, like this:

```
@RequestMapping(
  value = "/employees",
  produces = "application/json",
  method = RequestMethod.GET)
public List<Employee> findEmployees() {
    ...
}
```

```
// send the PUT request using CURL
$ curl --request PUT http://localhost:8080/api/employees
{"timestamp":1539720588712,"status":405,"error":"Method Not Allowed",
"exception":"org.springframework.web.HttpRequestMethodNotSupportedException",
"message":"Request method 'PUT' not supported","path":"/api/employees"}
```

## 5. 405 Not Support – Reason, Solution

What we're getting in this previous scenario is the HTTP response with the 405 Status Code –a client error that indicates that the server doesn't support the method/verb sent in the request.

As the name suggests here, the reason for this error is sending the request with a non-supported method.

As you can expect, we can solve this by defining an explicit mapping for PUT, in the existing method mapping:

```
@RequestMapping(
  value = "/employees",
  produces = "application/json",
  method = {RequestMethod.GET, RequestMethod.PUT}) ...
```

Alternatively, we can define the new method/mapping separately:

```
@RequestMapping(value = "/employees",
  produces = "application/json",
  method=RequestMethod.PUT)
public List<Employee> postEmployees() ...
```

## 6. Conclusion

The request method/verb is a critical aspect in HTTP communication, and we need to be careful with the exact semantics of the operations we define on the server side, and then with the exact requests we're sending in.

And as always, the examples shown in this article are available onover on GitHub.



Comments are closed on this article!

## CATEGORIES

SPRING
REST
JAVA
SECURITY
PERSISTENCE
JACKSON
HTTP CLIENT-SIDE

## SERIES

JAVA "BACK TO BASICS" TUTORIAL
JACKSON JSON TUTORIAL
HTTPCLIENT 4 TUTORIAL
REST WITH SPRING TUTORIAL
SPRING PERSISTENCE TUTORIAL
SECURITY WITH SPRING

## ABOUT

ABOUT BAELDUNG
THE COURSES
JOBS
THE FULL ARCHIVE
WRITE FOR BAELDUNG
EDITORS
OUR PARTNERS
ADVERTISE ON BAELDUNG

TERMS OF SERVICE
PRIVACY POLICY
COMPANY INFO
CONTACT