(/)

Spring's RequestBody and ResponseBody Annotations

Last updated: February 1, 2024

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

Reviewed by: Zeger Hendrikse (https://www.baeldung.com/editor/zeger-author)

REST (https://www.baeldung.com/category/rest)

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

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

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

1. Introduction

In this quick tutorial, we provide a concise overview of the Spring *@RequestBody* and *@ResponseBody* annotations.

Further reading:

Guide to Spring Handler Mappings (/spring-handler-mappings)

The article explains how HandlerMapping implementation resolve URL to a particular Handler.

Read more (/spring-handler-mappings) →

Quick Guide to Spring Controllers (/spring-controllers)

A quick and practical guide to Spring Controllers - both for typical MVC apps and for REST APIs.

Read more (/spring-controllers) →

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. *@RequestBody*

Simply put, **the *@RequestBody* annotation maps the *HttpRequest* body to a transfer or domain object, enabling automatic deserialization** of the inbound *HttpRequest* body onto a Java object.

First, let's have a look at a Spring controller method:

```
@PostMapping("/request")
public ResponseEntity postController(
  @RequestBody LoginForm loginForm) {

    exampleService.fakeAuthenticate(loginForm);
    return ResponseEntity.ok(HttpStatus.OK);
}
```

Spring automatically deserializes the JSON into a Java type, assuming an appropriate one is specified.

By default, **the type we annotate with the *@RequestBody* annotation must correspond to the JSON sent from our client-side controller:**

```
public class LoginForm {
    private String username;
    private String password;
    // ...
}
```

Here, the object we use to represent the *HttpRequest* body maps to our *LoginForm* object.

Let's test this using CURL:

(/)

```
curl -i \
-H "Accept: application/json" \
-H "Content-Type:application/json" \
-X POST --data
  '{"username": "johnny", "password": "password"}'
"https://localhost:8080/spring-boot-rest/post/request"
```

This is all we need for a Spring REST API and an Angular client using the *@RequestBody* annotation.

3. *@ResponseBody*

The *@ResponseBody* annotation tells a controller that the object returned is automatically serialized into JSON and passed back into the *HttpResponse* object.

Suppose we have a custom *Response* object:

```
public class ResponseTransfer {
    private String text;

    // standard getters/setters
}
```

Next, the associated controller can be implemented:

```
@Controller
@RequestMapping("/post")
public class ExamplePostController {

    @Autowired
    ExampleService exampleService;

    @PostMapping("/response")
    @ResponseBody
    public ResponseTransfer postResponseController(
      @RequestBody LoginForm loginForm) {
        return new ResponseTransfer("Thanks For Posting!!!");
     }
}
```

In the developer console of our browser or using a tool like Postman, we can see the following response:

```
{"text":"Thanks For Posting!!!"}
```

Remember, we don't need to annotate the *@RestController*-annotated controllers with the *@ResponseBody* annotation since Spring does it by default.

3.1. Setting the Content Type

When we use the *@ResponseBody* annotation, we're still able to explicitly set the content type that our method returns.

For that, **we can use the *@RequestMapping*'s *produces* attribute.** Note that annotations like *@PostMapping*, *@GetMapping*, etc. define aliases for that parameter.

Let's now add a new endpoint that sends a JSON response:

```
@PostMapping(value = "/content", produces =
MediaType.APPLICATION_JSON_VALUE)
@ResponseBody
public ResponseTransfer postResponseJsonContent(
  @RequestBody LoginForm loginForm) {
    return new ResponseTransfer("JSON Content!");
}
```

In the example, we used the *MediaType.APPLICATION_JSON_VALUE* constant. Alternatively, we can use *application/json* directly.

Next, let's implement a new method, mapped to the same */content* path, but returning XML content instead:

```
@PostMapping(value = "/content", produces =
MediaType.APPLICATION_XML_VALUE)
@ResponseBody
public ResponseTransfer postResponseXmlContent(
  @RequestBody LoginForm loginForm) {
    return new ResponseTransfer("XML Content!");
}
```

Now, **depending on the value of an *Accept* parameter sent in the request's header, we'll get different responses.**

Let's see this in action:

```
curl -i \
-H "Accept: application/json" \
-H "Content-Type:application/json" \
-X POST --data
  '{"username": "johnny", "password": "password"}'
"https://localhost:8080/spring-boot-rest/post/content"
```

The CURL command returns a JSON response:

```
HTTP/1.1 200
Content-Type: application/json
Transfer-Encoding: chunked
Date: Thu, 20 Feb 2020 19:43:06 GMT

{"text":"JSON Content!"}
```

Now, let's change the *Accept* parameter:

```
curl -i \
-H "Accept: application/xml" \
-H "Content-Type:application/json" \
-X POST --data
  '{"username": "johnny", "password": "password"}'
"https://localhost:8080/spring-boot-rest/post/content"
```

As anticipated, we get an XML content this time:

```
HTTP/1.1 200
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Thu, 20 Feb 2020 19:43:19 GMT

<ResponseTransfer><text>XML Content!</text></ResponseTransfer>
```

4. Conclusion

We've built a simple Angular client for the Spring app that demonstrates how to use the *@RequestBody* and *@ResponseBody* annotations.

Additionally, we showed how to set a content type when using *@ResponseBody*.

As always, code samples are available over on GitHub (https://github.com/eugenp/tutorials/tree/master/spring-boot-rest).

Comments are closed on this article!

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)