Spring Boot @ResponseStatus

Spring Boot @ResponseStatus tutorial shows how to use @ResponseStatus annotation in a Spring application.

Tweet

Spring is a popular Java application framework and *Spring Boot* is an evolution of Spring that helps create stand-alone, production-grade Spring based applications easily.

@ResponseStatus

`@ResponseStatus` marks a method or exception class with the status code and reason message that should be returned. The status code is applied to the HTTP response when the handler method is invoked, or whenever the specified exception is thrown. It overrides status information set by other means, like `ResponseEntity` or `redirect:`.

Spring Boot @ResponseStatus example

In the following application, we demonstrate the usage of the `@ResponseStatus` annotation. The applicati simulates a form for retrieving orders by their Id. Trying to find orders with an Id greater than 500 will throw an exception. As a consequence of this exception, a custom error page is displayed.

```
pom.xml
src
├── main
│   ├── java
│   │   └── com
│   │       └── zetcode
│   │           ├── Application.java
│   │           ├── controller
│   │           │   └── MyController.java
│   │           └── exception
│   │               └── OrderNotFoundException.java
│   └── resources
│       ├── static
│       │   └── index.html
│       └── templates
│           └── error.ftl
```

```
└── test
    └── java
```

This is the project structure of the Spring application.

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>
    <groupId>com.zetcode</groupId>
    <artifactId>springbootresponsestatusex</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>11</maven.compiler.source>
        <maven.compiler.target>11</maven.compiler.target>
    </properties>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.1.1.RELEASE</version>
    </parent>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-freemarker</artifactId>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

This is the Maven `pom.xml` file. The `spring-boot-starter-freemarker` is a dependency for Freemarker template engine; the `spring-boot-maven-plugin` packages Spring applications into executable JAR or

WAR archives.

com/zetcode/controller/MyController.java

```
package com.zetcode.controller;

import com.zetcode.exception.OrderNotFoundException;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
public class MyController {

    @RequestMapping(value = "/orders/{id}")
    @ResponseBody
    public String getOrder(@PathVariable("id") long id) {

        if (id < 0 || id > 500) {

            var message = String.format("Order %d not found", id);
            throw new OrderNotFoundException(message);
        }

        var message = String.format("Returning order %d", id);

        return message;
    }
}
```

The `MyController's getOrder()` method responds to the client request. It reads the order Id from the path with `@PathVariable`.

```
if (id < 0 || id > 500) {

    var message = String.format("Order %d not found", id);
    throw new OrderNotFoundException(message);
}
```

For invalid orders (id < 0) and orders greater than 500, we throw the `OrderNotFoundException` exceptio This is a simple simulation of an order system.

```
var message = String.format("Returning order %d", id);
```

For other orders Ids, we return a message indicating that the order was found and returned.

com/zetcode/exception/OrderNotFoundException.java

```
package com.zetcode.exception;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;
```

```
@ResponseStatus(value = HttpStatus.NOT_FOUND, reason = "No such order")
public class OrderNotFoundException extends RuntimeException {

    public OrderNotFoundException(String message) {

        super(message);
    }
}
```

We have a custom `OrderNotFoundException`. It is decorated with the `@ResponseStatus` annotation. The value is set to `HttpStatus.NOT_FOUND` and the reason message says "No such order". This information wil be used in the error page.

resources/static/index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Home page</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>
    <body>
        <a href="/orders/505/">Get order with Id 505</a>
    </body>
</html>
```

This is the home page. It contains a link that looks for an order with Id 505. The file is located in the `src/main/resources/static` directory.

resources/templates/error.ftl

```
<html>
    <head>
        <title>Error page</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>

    <body>
        <div>
            <h1>Error occurred</h1>

            <p>${status}: ${error} - ${message}</p>
        </div>
    </body>
</html>
```

The `error.ftl` is a generic error page. It is Freemarker template file which shows status, error, and the reason message. These values were set with `@ResponseStatus` earlier. It is located in the `src/main/resources/templates` directory.

com/zetcode/Application.java

```
package com.zetcode;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class Application  {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }
}
```

`Application` is the entry point which sets up Spring Boot application.

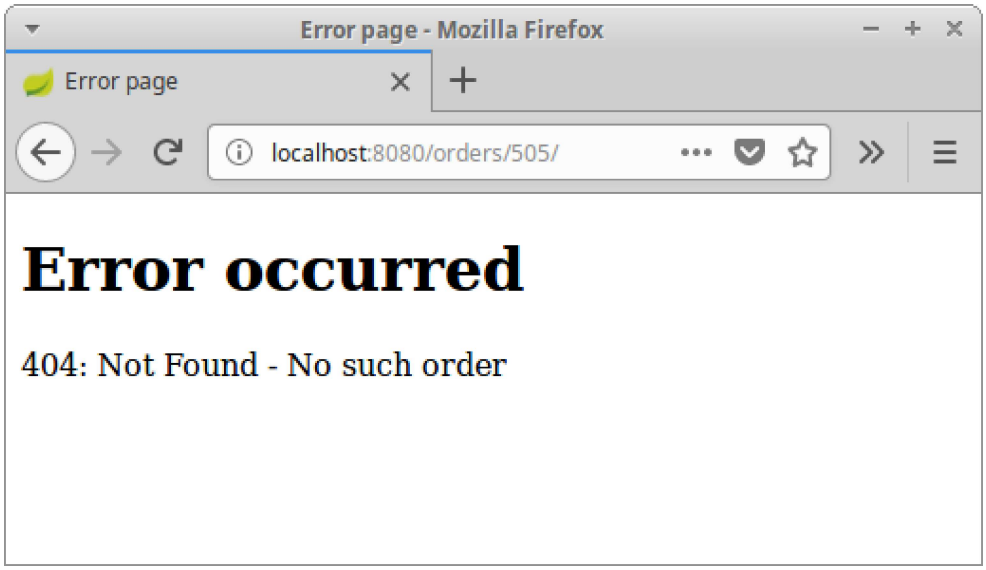

Figure: Error page

In this tutorial, we have shown how to use the `@ResponseStatus` annotation in a Spring application. You might also be interested in the related tutorials: Spring Boot ResponseEntity tutorial, Spring Boot @ExceptionHandler tutorial, Java tutorial, or list all Spring Boot tutorials.

Home Top of Page