How to Validate Request Parameters in Spring Boot

In the Spring Boot MVC, the url contains query parameters. These query parameters are identified as request parameters. The request url contains query parameters. The spring boot identifies a request parameter in the controller class. Within this post, we'll see "how to read request parameter in spring boot" and how the request parameters within the spring boot are validated.

The request parameters are the values sent from the client side. The server will read the values as a request parameters. These values are read only in nature. Any value that has to be sent to the client are assigned in the response object.

Spring boot supports validation of the request parameter before processing in the controller class. Here, we will see the step-by-step procedure to How to configure request parameter in spring boot, How to read request parameter in spring boot and how to validate the request parameters in spring boot.

Step 1 – Add dependency in pom.xml

The "spring-boot-starter-validation" dependency will include all the jars needed for the validation of the request parameters. The web application needs the dependence of "spring-boot-starter-browser." This will allow the tomcat server and start as a web application. The default "spring-boot-starter-test" test dependency will be applied to pom.xml for the unit testing.

If you have created a spring boot application, now you're trying to incorporate the validation code, then add the "spring-boot-starter-validation" dependency to the pom.xml file. The complete pom.xml file will be as shown below.

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 https://maven.apache.org/xsd/maven-
4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>2.2.5.RELEASE</version>
                <relativePath /> <!-- lookup parent from repository -->
        </parent>
        <groupId>com.yawintutor</groupId>
        <artifactId>Spring-Application</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>SpringBootValidation</name>
        <description>Spring Boot Project</description>

        <properties>
                <java.version>1.8</java.version>
        </properties>

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

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

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

</project>
```

Step 2 – Add @RequestParam annotation

The query param is identified as the request param in the controller class with an @RequestParam annotation. The request params will be a ssigned to the query param value automatically at the start of the spring. If the query param is not set, the request param is set as null or by default value.

```
package com.yawintutor;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class TestController {

	@GetMapping("/student/")
	ResponseEntity<String> studentR(@RequestParam("age") int age) {
		return ResponseEntity.ok("Your age is " + age);
	}
}
```

Step 3 – Add @Validated annotation for validation

The @Validated annotation is used to specify the rest controller to be validated. The spring boot application validates all the method before it is called. @Min(5) is added in this case, to validate the request parameters in the example below.

TestController.java

```
package com.yawintutor;

import javax.validation.constraints.Min;

import org.springframework.http.ResponseEntity;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
@Validated
public class TestController {

	@GetMapping("/student/")
	ResponseEntity<String> studentR(@RequestParam("age") @Min(5) int age) {
		return ResponseEntity.ok("Your age is " + age);
	}
}
```

Step 4 – Run the application

To validate the query parameter, start the application on the spring boot. Call the url below that is executing without error.

```
http://localhost:8080/student/?age=5
```

Output

```
Your age is 5
```

If the invalid input is passed as part of url the validation will fail. The url below displays the request parameter with invalid data.

```
http://localhost:8080/student/?age=2
```

Output

```
Whitelabel Error Page
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Wed Mar 25 08:29:24 IST 2020
There was an unexpected error (type=Internal Server Error, status=500).
student.age: must be greater than or equal to 5
```

Exception

The exception below will appear either in the console window or in the log file.

```
2020-03-25 08:29:24.577 ERROR 88142 --- [nio-8080-exec-1] o.a.c.c.C.[.[.[/].[dispatcherServlet]    :
Servlet.service() for servlet [dispatcherServlet] in context with path [] threw exception [Request
processing failed; nested exception is javax.validation.ConstraintViolationException: student.age: must be
greater than or equal to 5] with root cause

javax.validation.ConstraintViolationException: student.age: must be greater than or equal to 5
```

How to view the request url for debugging

To understand the exception, It is important to know what the url is called. Unable to see or log the actual value inside the method. Because the validation fails before calling the method. The best way is to allow debug logging by adding the code below in application.properties file.

application.properties

```
logging.level.org.springframework.web.servlet.DispatcherServlet=debug
```

The log will look as below

```
2020-03-25 08:29:24.499 DEBUG 88142 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : GET
"/student/?age=2", parameters={}
2020-03-25 08:29:24.572 DEBUG 88142 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Failed
to complete request: javax.validation.ConstraintViolationException: student.age: must be greater than or
equal to 5
```

Other related Links

How to Validate Path Variable in Spring Boot
How to Validate Request Parameters in Spring Boot
How to Validate Request Body in Spring Boot
How to Validate Request Headers in Spring Boot
How to Validate Spring Boot Bean Programmatically or Manually
How to Customize Default Error Message in Spring Boot Validation
How to Customize Default Error Message using @ControllerAdvice in Spring Boot Validation