tutorialspoint.com

Spring Boot - Exception Handling

8–10 minutes

Handling exceptions and errors in APIs and sending the proper response to the client is good for enterprise applications. In this chapter, we will learn how to handle exceptions in Spring Boot.

Before proceeding with exception handling, let us gain an understanding on the following annotations.

Controller Advice

The @ControllerAdvice is an annotation, to handle the exceptions globally.

Exception Handler

The @ExceptionHandler is an annotation used to handle the specific exceptions and sending the custom responses to the client.

You can use the following code to create @ControllerAdvice class to handle the exceptions globally −

```
package com.tutorialspoint.demo.exception;

import org.springframework.web.bind.annotation.ControllerAdvice;

@ControllerAdvice
  public class ProductExceptionController {
}
```

Define a class that extends the RuntimeException class.

```
package com.tutorialspoint.demo.exception;

public class ProductNotfoundException extends RuntimeException
{
  private static final long serialVersionUID = 1L;
}
```

You can define the @ExceptionHandler method to handle the exceptions as shown. This method should be used for writing the Controller Advice class file.

```
@ExceptionHandler(value = ProductNotfoundException.class)
```

```
public ResponseEntity<Object>
exception(ProductNotfoundException exception) {
}
```

Now, use the code given below to throw the exception from the API.

```
@RequestMapping(value = "/products/{id}", method =
RequestMethod.PUT)
public ResponseEntity<Object> updateProduct() {
  throw new ProductNotfoundException();
}
```

The complete code to handle the exception is given below. In this example, we used the PUT API to update the product. Here, while updating the product, if the product is not found, then return the response error message as “Product not found”. Note that the **ProductNotFoundException** exception class should extend the **RuntimeException**.

```
package com.tutorialspoint.demo.exception;
public class ProductNotfoundException extends RuntimeException
{
  private static final long serialVersionUID = 1L;
}
```

The Controller Advice class to handle the exception globally is given below. We can define any Exception Handler methods in this class file.

```
package com.tutorialspoint.demo.exception;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;

@ControllerAdvice
public class ProductExceptionController {
  @ExceptionHandler(value = ProductNotfoundException.class)
  public ResponseEntity<Object>
exception(ProductNotfoundException exception) {
    return new ResponseEntity<>("Product not found",
HttpStatus.NOT_FOUND);
  }
}
```

The Product Service API controller file is given below to update the Product. If the Product is not found, then it throws the **ProductNotFoundException** class.

```
package com.tutorialspoint.demo.controller;

import java.util.HashMap;
```

```
import java.util.Map;

import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import
com.tutorialspoint.demo.exception.ProductNotfoundException;
import com.tutorialspoint.demo.model.Product;

@RestController
public class ProductServiceController {
  private static Map<String, Product> productRepo = new
HashMap<>();
  static {
    Product honey = new Product();
    honey.setId("1");
    honey.setName("Honey");
    productRepo.put(honey.getId(), honey);

    Product almond = new Product();
    almond.setId("2");
    almond.setName("Almond");
    productRepo.put(almond.getId(), almond);
  }

  @RequestMapping(value = "/products/{id}", method =
RequestMethod.PUT)
  public ResponseEntity<Object>
updateProduct(@PathVariable("id") String id, @RequestBody
Product product) {
    if(!productRepo.containsKey(id))throw new
ProductNotfoundException();
    productRepo.remove(id);
    product.setId(id);
    productRepo.put(id, product);
    return new ResponseEntity<>("Product is updated
successfully", HttpStatus.OK);
  }
}
```

The code for main Spring Boot application class file is given below
–

```
package com.tutorialspoint.demo;
```

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

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

The code for **POJO class** for Product is given below −

```
package com.tutorialspoint.demo.model;
public class Product {
  private String id;
  private String name;

  public String getId() {
    return id;
  }
  public void setId(String id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
}
```

The code for **Maven build – pom.xml** is shown below −

```
<?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.tutorialspoint</groupId>
  <artifactId>demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>demo</name>
  <description>Demo project for Spring Boot</description>

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
```

```
    <version>1.5.8.RELEASE</version>
    <relativePath/>
  </parent>

  <properties>
    <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-
8</project.reporting.outputEncoding>
    <java.version>1.8</java.version>
  </properties>

  <dependencies>
    <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>
```

The code for **Gradle Build – build.gradle** is given below −

```
buildscript {
  ext {
    springBootVersion = '1.5.8.RELEASE'
  }
  repositories {
    mavenCentral()
  }
  dependencies {
    classpath("org.springframework.boot:spring-boot-gradle-
plugin:${springBootVersion}")
  }
}
apply plugin: 'java'
```

```
apply plugin: 'eclipse'
apply plugin: 'org.springframework.boot'

group = 'com.tutorialspoint'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = 1.8

repositories {
  mavenCentral()
}
dependencies {
  compile('org.springframework.boot:spring-boot-starter-web')
  testCompile('org.springframework.boot:spring-boot-starter-test')
}
```

You can create an executable JAR file, and run the Spring Boot application by using the Maven or Gradle commands −

For Maven, you can use the following command −

```
mvn clean install
```

After “BUILD SUCCESS”, you can find the JAR file under the target directory.

For Gradle, you can use the following command −

```
gradle clean build
```

After “BUILD SUCCESSFUL”, you can find the JAR file under the build/libs directory.

You can run the JAR file by using the following command −

```
java –jar <JARFILE>
```

This will start the application on the Tomcat port 8080 as shown below −

Now hit the below URL in POSTMAN application and you can see the output as shown below −

Update URL: http://localhost:8080/products/3