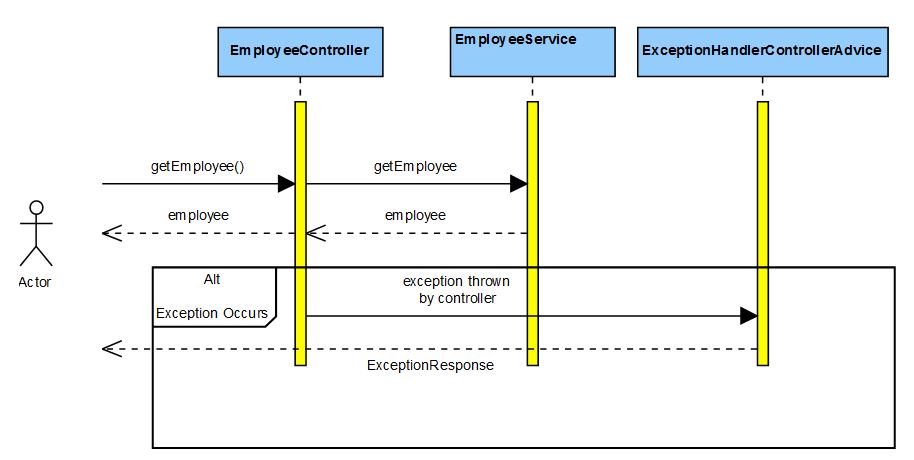
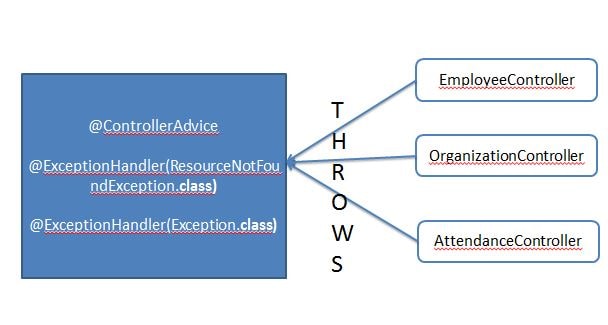
**Spring Boot Exception Handling Using ControllerAdvice**

Spring provides a very useful way to handle exceptions using ControllerAdvice.  
We will be implementing a ControlerAdvice class which will handle all exceptions thrown by the controller class.  
  
Exceptions thrown by a Controller method is mapped to the ControllerAdvice method using @ExceptionHandler annotations.  
  
  
Maven Project will be as follows-  
  
In the Maven we need the spring boot web dependency.Maven will be as follows-

<?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.javainuse</groupId>

<artifactId>spring-boot-exceptionhandling</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>SpringBootHelloWorld</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.4.1.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</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>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

Create the SpringBootHelloWorldApplication.java as below-

package com.javainuse;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringBootHelloWorldApplication {

public static void main(String[] args) {

SpringApplication.run(SpringBootHelloWorldApplication.class, args);

}

}

Create the Employee model class as follows-

package com.javainuse.model;

public class Employee {

private String empId;

private String name;

private String designation;

private double salary;

public Employee() {

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDesignation() {

return designation;

}

public void setDesignation(String designation) {

this.designation = designation;

}

public double getSalary() {

return salary;

}

public void setSalary(double salary) {

this.salary = salary;

}

public String getEmpId() {

return empId;

}

public void setEmpId(String empId) {

this.empId = empId;

}

}

We first define the classes in the service package.  
Define the EmployeeServiceException to thrown by the EmployeeService class in case of exceptionss.

package com.javainuse.service;

public class EmployeeServiceException extends Exception {

private static final long serialVersionUID = -470180507998010368L;

public EmployeeServiceException() {

super();

}

public EmployeeServiceException(final String message) {

super(message);

}

}

Define the service class. We have defined three methods in it for explaining 3 scenarios.

package com.javainuse.service;

import org.springframework.stereotype.Service;

import com.javainuse.model.Employee;

@Service

public class EmployeeService {

**//return employee object**

public Employee getEmployee() throws EmployeeServiceException {

Employee emp = new Employee();

emp.setName("emp1");

emp.setDesignation("manager");

emp.setEmpId("1");

emp.setSalary(3000);

return emp;

}

**//return employee as null**

public Employee getEmployeeNull() throws EmployeeServiceException {

return null;

}

**//throw exception**

public Employee getEmployeeException() throws EmployeeServiceException {

throw new EmployeeServiceException();

}

}

Now we define the Controller package classes  
Define the ResourceNotFoundException class. This Exception will be thrown by the controller when no resource i.e. employee to be returned in our case is found.

package com.javainuse.exceptionhandling;

public class ResourceNotFoundException extends Exception {

private static final long serialVersionUID = -9079454849611061074L;

public ResourceNotFoundException() {

super();

}

public ResourceNotFoundException(final String message) {

super(message);

}

}

Next we define the controller class  
@RequestMapping maps /employee request to return an employee object. We have two other requestmappings, one returns a null employee and the other throws an exception. Also the EmployeeController has the EmployeeService instance autowired.

package com.javainuse.controller;

import org.springframework.beans.factory.annotation.Autowired;

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

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

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

import com.javainuse.exceptionhandling.ResourceNotFoundException;

import com.javainuse.model.Employee;

import com.javainuse.service.EmployeeService;

import com.javainuse.service.EmployeeServiceException;

@RestController

public class EmployeeController {

@Autowired

EmployeeService employeeService;

**//Happy path, an employee is returned as response**

@RequestMapping(value = "/employee", method = RequestMethod.GET)

public Employee getEmpl() throws ResourceNotFoundException, EmployeeServiceException {

try {

Employee emp = employeeService.getEmployee();

if (emp == null) {

throw new ResourceNotFoundException("Employee not found");

}

return emp;

} catch (EmployeeServiceException e) {

throw new EmployeeServiceException("Internal Server Exception while getting exception");

}

}

**//no employee found so ResourceNotFoundException is thrown**

@RequestMapping(value = "/employee2", method = RequestMethod.GET)

public Employee getEmp2() throws ResourceNotFoundException, EmployeeServiceException {

try {

Employee emp = employeeService.getEmployeeNull();

if (emp == null) {

throw new ResourceNotFoundException("Employee not found");

}

return emp;

} catch (EmployeeServiceException e) {

throw new EmployeeServiceException("Internal Server Exception while getting exception");

}

}

**//Some exception is thrown by service layer**

@RequestMapping(value = "/employee3", method = RequestMethod.GET)

public Employee getEmp3() throws ResourceNotFoundException, EmployeeServiceException {

try {

Employee emp = employeeService.getEmployeeException();

if (emp == null) {

throw new ResourceNotFoundException("Employee not found");

}

return emp;

} catch (EmployeeServiceException e) {

throw new EmployeeServiceException("Internal Server Exception while getting exception");

}

}

}

Define the POJO class which will be sent as response by the ControllerAdvice class.

package com.javainuse.exceptionhandling;

public class ExceptionResponse {

private String errorMessage;

private String requestedURI;

public String getErrorMessage() {

return errorMessage;

}

public void setErrorMessage(final String errorMessage) {

this.errorMessage = errorMessage;

}

public String getRequestedURI() {

return requestedURI;

}

public void callerURL(final String requestedURI) {

this.requestedURI = requestedURI;

}

}

The ExceptionHandlerControllerAdvice will catch the exceptions thrown by the controller method, and we send more appropriate response to the caller. For example if an exception occurs we don't want the caller to get the entire stack trace involving technical details. **Also we can return the appropriate response status depending on the business logic.**

package com.javainuse.exceptionhandling;

import javax.servlet.http.HttpServletRequest;

import org.springframework.http.HttpStatus;

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

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

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

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

@ControllerAdvice

public class ExceptionHandlerControllerAdvice {

@ExceptionHandler(ResourceNotFoundException.class)

@ResponseStatus(value = HttpStatus.NOT\_FOUND)

public @ResponseBody ExceptionResponse handleResourceNotFound(final ResourceNotFoundException exception,

final HttpServletRequest request) {

ExceptionResponse error = new ExceptionResponse();

error.setErrorMessage(exception.getMessage());

error.callerURL(request.getRequestURI());

return error;

}

@ExceptionHandler(Exception.class)

@ResponseStatus(value = HttpStatus.INTERNAL\_SERVER\_ERROR)

public @ResponseBody ExceptionResponse handleException(final Exception exception,

final HttpServletRequest request) {

ExceptionResponse error = new ExceptionResponse();

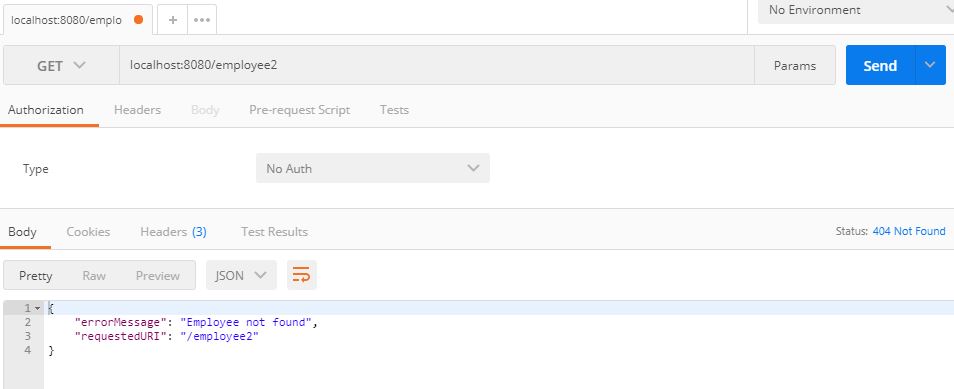
error.setErrorMessage(exception.getMessage());

error.callerURL(request.getRequestURI());

return error;

}

}

Compile and the run the SpringBootHelloWorldApplication.java as a Java application.  
Using postman go to **localhost:8080/employee2** Employee is not found, so ResourceNotFoundException is thrown. **Response code is sent as 404.**  
  
**localhost:8080/employee3** EmployeeServiceException is thrown. **Response code is sent as 500.**  
