**Assisted Practice: 2.1 Spring Boot Exception Handling**

This section will guide you to:

* Set up Eclipse to work with Spring Boot using the STS plugin
* Create an entity class to store product data
* Create a custom Exception class for products
* Create a Controller to handle the custom Exception
* Create a Controller class to throw the custom Exception

**Development Environment**

* Eclipse IDE for Enterprise Java Developers v2019-03 (4.11.0)
* Apache Tomcat Server v9.0
* JRE: OpenJDK Runtime Environment 11.0.2
* Spring Boot STS 4
* All other software components are configured automatically by Spring Boot

This lab has ten subsections, namely:

* + 1. Installing the STS Plugin in Eclipse
    2. Creating a Spring Boot Starter Project which is web enabled
    3. Creating a EProduct entity class
    4. Creating a ProductNotFoundException class
    5. Creating a EProductExceptionController class
    6. Creating MainController to throw ProductNotFoundException
    7. Building the project
    8. Publishing and starting the project
    9. Running the project
    10. Pushing the code to your GitHub repositories

**Step 2.1.1:** Installing the Spring Tool Suite Plugin in Eclipse

* Spring Tool Suite is already installed as an Eclipse plugin in your practice lab. (Refer FSD: Lab Guide - Phase 3 to verify the installation.)

**Step 2.1.2:** Creating a Spring Boot Starter Project which is web enabled

* Open Eclipse
* Go to the **File** menu. Choose **New->Other**
* In the **Wizards** list, select **Spring Boot->Spring Starter Project**
* In **Name,** enter SpringBootStarter, **Type** as Maven, **Packaging** as Jar, **Group** as com.ecommerce, and **Package** as com.ecommerce
* Click on **Next**
* In the list of **Available** dependencies, scroll down to select **Web->Spring Web Starter**
* Click on **Next**
* Click on **Finish**
* This will create the project files in the Project Explorer

**Step 2.1.3:** Creating a EProduct entity class

* In the Project Explorer, expand **SpringBootStarter->src->main>java>com->ecommerce**
* Right click on ecommerce and click on **New->Folder**
* Enter **Folder Name** as entity and click on **Finish**
* Right click on **entity** and click on **New->Other**
* In the **Wizard** list, choose **Class** and click on **Next**
* In **Name,** enter EProduct and click on **Finish**
* Add the following code:

**package** com.ecommerce.entity;

**import** java.math.BigDecimal;

**import** java.util.Collection;

**import** java.util.Date;

**import** java.util.List;

**import** java.util.Set;

**import** java.util.Map;

**public** **class** EProduct {

**private** long ID;

**private** **String** name;

**private** **BigDecimal** price;

**private** **Date** dateAdded;

**public** EProduct() {

}

**public** long getID() {**return** **this**.ID; }

**public** **String** getName() { **return** **this**.name;}

**public** **BigDecimal** getPrice() { **return** **this**.price;}

**public** **Date** getDateAdded() { **return** **this**.dateAdded;}

**public** void setID(long id) { **this**.ID = id;}

**public** void setName(**String** name) { **this**.name = name;}

**public** void setPrice(**BigDecimal** price) { **this**.price = price;}

**public** void setDateAdded(**Date** date) { **this**.dateAdded = date;}

}

**Step 2.1.4:** Creating a ProductNotFoundException class

* In the Project Explorer, expand **SpringBootStarter->src->main>java>com->ecommerce**
* Right click on **ecommerce** and click on **New->Folder**
* Enter **Folder Name** as **exceptions** and click on **Finish**
* Right click on **exceptions** and click on **New->Other**
* In the **Wizard** list, choose **Class** and click on **Next**
* In **Name,** enter **ProductNotFoundException** and click on **Finish**
* Add the following code:

**package** com.ecommerce.exceptions;

**public** **class** ProductNotFoundException **extends** RuntimeException {

**private** **static** **final** long serialVersionUID = 1L;

}

**Step 2.1.5:** Creating a EProductExceptionController class

* In the Project Explorer, expand **SpringBootStarter->src->main>java>com->ecommerce**
* Right click on **ecommerce** and click on **New->Folder**
* Enter **Folder Name** as **controllers** and click on **Finish**
* Right click on **controllers** and click on **New->Other**
* In the **Wizard** list, choose **Class** and click on **Next**
* In **Name,** enter **EProductExceptionController** and click on **Finish**
* Add the following code:

**package** com.ecommece.contrrollers;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

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

**import** com.ecommerce.exceptions.ProductNotFoundException;

**@ControllerAdvice**

**public** **class** EProductExceptionController {

**@ExceptionHandler(value = ProductNotFoundException.class)**

**public** ResponseEntity<**Object**> exception(ProductNotFoundException exception) {

**return** **new** ResponseEntity<>("Product not found", HttpStatus.NOT\_FOUND);

}

}

**Step 2.1.6:** Creating MainController to throw ProductNotFoundException

* In the Project Explorer, expand **SpringBootStarter->src->main>java>com->ecommerce->controllers**
* Right click on **controllers** and click on **New->Other**
* In the **Wizard** list, choose **Class** and click on **Next**
* In **Name,** enter **MainController** and click on **Finish**
* Add the following code:

**package** com.ecommerce.controllers;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.stereotype.Controller;

**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.ResponseBody;

**import** com.ecommerce.entity.EProduct;

**import** com.ecommerce.exceptions.ProductNotFoundException;

**@Controller**

**public** **class** MainController {

**@RequestMapping(value = "/product/{id}", method = RequestMethod.GET)**

**@ResponseBody**

**public** **String** getProduct(**@PathVariable("id")** **String** id) {

**if** (id.contentEquals("0"))

**throw** **new** ProductNotFoundException();

**return** "Product was found";

}

}

**Step 2.1.7:** Building the project

* From the **Project** menu at the top, click on **Build**
* If any compile errors are shown, fix them as required

**Step 2.1.8:** Publishing and starting the project

* In the Project Explorer, right click on **SpringBootStarter->Run As->Spring Boot App**
* Check in the Eclipse Console for the message **Started SpringBootStarterApplication**

**Step 2.1.9:** Running the project

* To run the project, open a web browser and type [**http://localhost:8080/product/0**](http://localhost:8080/product/0)to throw the Exception
* Type [**http://localhost:8080/product/1**](http://localhost:8080/product/1)to see output from the MainController without throwing an Exception

**Step 2.1.10:** Pushing the code to your GitHub repositories

* Open your command prompt and navigate to the folder where you have created your files.

**cd <folder path>**

* Initialize your repository using the following command:

**git init**

* Add all the files to your git repository using the following command:

**git add .**

* Commit the changes using the following command:

**git commit . -m “Changes have been committed.”**

* Push the files to the folder you initially created using the following command:

**git push -u origin master**