

Farm Fresh Online store - V1

Grade settings: Maximum grade: 100

Based on: [Farm Fresh Online store - V1](#)

Run: Yes **Evaluate:** Yes

Automatic grade: Yes **Maximum execution time:** 240 s **Maximum memory used:** 1.50

GiB **Maximum execution file size:** 1.50 GiB **Maximum number of processes:** 10000

Farm-Fresh online store

FarmFresh-A natural farm store, sells organic farm fresh fruits through online, they approach you to create a separate online application for calculating total bill amount for the order placed by the customer.

Create a Spring MVC Spring Boot Application for developing an Online Application. Design a showPage to choose an appropriate product(**Apple/Mango/Orange/Grapes/JackFruit**)and enter the quantity in kg.

On clicking CalculateCost button, the application should calculate the total bill amount based on the product chosen for the quantity. The customer has to then be redirected to billDesk.jsp page that displays the product chosen and its cost.

Application Work Flow:

- **ShoppingController** is the Controller class.
- **Product** is the model class with 3 attributes productName, quantity, and costPerKg along with its getters and setters.
- **ShoppingService** is the Service class which has a method called **calculateCost** that takes **Product** as its argument and returns double.
- **ShoppingService** class has getProductStock() method. It return the Map <String,Integer>. The product and its quantity is shown in the below table

productName	Quantity
Apple	50
Orange	30
JackFruit	25
Mango	75
Grapes	10

o Compare the quantity entered by the user with the quantity in the Map for the relevant product and check if the needed quantity is available. If available, then calculate the bill amount. Else throw a user defined exception NoStockException with a message **“No enough stock for product <<productName>>”**

o This method should set the value for **costPerKg** instance variable based on productName. To do this consider the below table.

Product Name	Cost Per Kg
Apple	250.00
Orange	90.00
JackFruit	75.00
Mango	60.00
Grapes	150.00

- o Output should be returned as double.

Initially, the customer should be routed via the **ShoppingController**'s **showPage** method to **showPage.jsp** that allows customer to choose the product and enter the quantity in kg.

[Note: showPage method has to be written inside the ShoppingController]

A method in the **ShoppingController** known as **populateProduct** should be annotated with the `ModelAttribute` `productNames`. This method should populate the product name (**Apple/Orange/Grapes/Mango/JackFruit**) in the `List<String>`. This should be then used to auto-populate the `productName` in the **showPage.jsp**

[Note:populateProduct method should be written inside the ShoppingController]

On clicking the **CalculateCost** button, the **ShoppingController**'s **calculateCost** method should be called. This method takes three arguments - model attribute named `product` which holds the form populated `Product`, `BindingResult` and the `ModelMap`.

- This method should calculate the `totalCost` by invoking the **calculateCost** method of the **ShoppingService**.
- `totalCost` has to be calculated as shown in the below example,

Example:

If customer choose product as `Orange` and quantity as 15, set `costPerKg=90.0`, Check the stock Map. For orange stock has the quantity of 30 kg.

$\text{cost} = \text{quantity} * \text{costPerKg} = 15 * 90 = 1350.0$

If customer choose product as `Grapes` and quantity as 25, then check the quantity in the stock map. For Grapes the quantity is 10Kg. So raise `NoStockException` with message **"No enough stock for product <<productName>>"**

- If the quantity entered by the customer is less than 1 then an error message **"Minimum Quantity should be 1Kg"** has to be displayed.

showPage.jsp

Farm Fresh - A natural farm store!!

Select product

Quantity

- If the quantity entered by the customer is less than 1 then an error message "Minimum quantity should be 1 Kg" has to be displayed.

Farm Fresh - A natural farm store!!

Select product

Quantity Minimum quantity should be 1Kg

Redirect the customer to billDesk.jsp and display the output as like in the below screenshot

billDesk.jsp

Farm Fresh - A natural farm store!!

Product Name Apple
Quantity 20
Cost Per Kg 250.0
Total cost Rs 5000.0

exceptionPage.jsp

Farm Fresh - A natural farm store!!

Select product

Quantity

Farm Fresh - A natural farm store!!

No enough stock for product Apple

[Home](#)

Design Constraints:

UI Design Constraints:

showPage.jsp		
Component	ID	Constraints
DropDown	productName	Should be auto-populated using the model attribute written above the populateProduct method inside the ShoppingController . Do not hard code the values
Textbox	quantity	Minimum quantity should be 1Kg
submit	submit	Both id and name attribute value should be "submit"

Component Specification

Controller

ShoppingController			
AttributeName	AttributeType	Access Specifier	Constraints
ShoppingService	ShoppingService	private	Use annotation to Autowire

ShoppingController			
Method Name	Method Argument name: type	Return type	RequestMapping URL & Request Method
showPage	ModelAttribute "product": Product	String	/showPage& GET
calculateCost	ModelAttribute "product": Product, result: BindingResult, model: ModelMap	String	/calculate& POST
populateProduct		List <String>	Should be annotated with ModelAttribute with name "productNames"
exceptionHandler	Exception exception		Use appropriate annotation to handle the exception

Service

ShoppingService		
Method Name	Method Argument name: type	Return type
calculateCost	product:Product	Double
getProductStock		Map<String,Integer>

Model

Product	
AttributeName	AttributeType
productName	String
costPerKg	double
quantity	int

Overall Design Constraints:

- **ShoppingController** should be inside the package **com.controller**
- **ShoppingService** should be inside the package **com.service**
- **Product** should be in the package **com.model**
- Use appropriate annotation to configure ShoppingService as a Service
- Use appropriate annotation to configure ShoppingController as a Controller
- **ShoppingService** should be autowired inside the **ShoppingController**.
- Use annotations to implement the business Validation as specified in the screen shot [That is, when the quantity is less than 1 then an error message "**Minimum quantity should be 1 Kg**" should be rendered in the UI]
- Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.
- In the pom.xml you are provided with all the dependencies needed for developing the application.
- You will not be evaluated based on the UI design (layout, color, formatting, etc.). You are free to have a basic UI with all the required UI components (input fields, buttons, labels, etc.). Expected components with the id alone should be designed as per the requirement.
- Adhere to the design specifications mentioned in the case study.

Â. Do not change or delete the class/method names or return types which are provided to you as a part of the base code skeleton.

Â. Please make sure that your code does not have any compilation errors while submitting your case study solution.